

Design of Spatially Coupled LDPC Codes over $\text{GF}(q)$ for Windowed Decoding

Lai Wei, *Student Member, IEEE*, David G. M. Mitchell, *Member, IEEE*,
Thomas E. Fuja, *Fellow, IEEE*, and Daniel J. Costello, Jr., *Life Fellow, IEEE*

Abstract

In this paper we consider the generalization of binary spatially coupled low-density parity-check (SC-LDPC) codes to finite fields $\text{GF}(q)$, $q \geq 2$, and develop design rules for q -ary SC-LDPC code ensembles based on their iterative belief propagation (BP) decoding thresholds, with particular emphasis on low-latency windowed decoding (WD). We consider transmission over both the binary erasure channel (BEC) and the binary-input additive white Gaussian noise channel (BIAWGNC) and present results for a variety of (J, K) -regular SC-LDPC code ensembles constructed over $\text{GF}(q)$ using protographs. Thresholds are calculated using protograph versions of q -ary density evolution (for the BEC) and q -ary extrinsic information transfer analysis (for the BIAWGNC). We show that WD of q -ary SC-LDPC codes provides significant threshold gains compared to corresponding (uncoupled) q -ary LDPC block code (LDPC-BC) ensembles when the window size W is large enough and that these gains increase as the finite field size $q = 2^m$ increases. Moreover, we demonstrate that the new design rules provide WD thresholds that are close to capacity, even when both m and W are relatively small (thereby reducing decoding complexity and latency). The analysis further shows that, compared to standard flooding-schedule decoding, WD of q -ary SC-LDPC code ensembles results in significant reductions in both decoding complexity and decoding latency, and that these reductions increase as m increases. For applications with a near-threshold performance requirement and a constraint on decoding latency, we show that using q -ary SC-LDPC code ensembles, with moderate $q > 2$, instead of their binary counterparts results in reduced decoding complexity.

This work was supported by the U.S. National Science Foundation under grant CCF-1161754. Some of the material in this paper was presented at the Information Theory and Applications Workshop, San Diego, CA, Feb. 2014, and at the IEEE International Symposium on Information Theory, Honolulu, HI, July 2014.

L. Wei, D. G. M. Mitchell, T. E. Fuja, and D. J. Costello, Jr. are with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN, 46556, U.S. (e-mail: {lwei1, david.mitchell, tfuja, dcostell1}@nd.edu).

Index Terms

q -ary spatially coupled low-density parity-check codes, protographs, edge spreading, iterative decoding thresholds, binary erasure channel, q -ary density evolution, binary-input additive white Gaussian noise channel, q -ary extrinsic information transfer analysis, flooding-schedule decoding, windowed decoding, decoding complexity, decoding latency

I. INTRODUCTION

Low-density parity-check block codes (LDPC-BCs) constructed over finite fields $\text{GF}(q)$ of size $q > 2$ outperform comparable binary LDPC-BCs [1], in particular when the block length is short to moderate. However, this performance gain comes at the cost of an increase in decoding complexity. A direct implementation of the q -ary belief-propagation (BP) decoder, originally proposed by Davey and MacKay in [1], has complexity $\mathcal{O}(q^2)$ per symbol. More recently, an implementation based on the fast Fourier transform [2] was shown to reduce the complexity to $\mathcal{O}(q \log q)$. Beyond that, a variety of simple but sub-optimal decoding algorithms have been proposed in the literature, such as the extended min-sum (EMS) algorithm [3] and the trellis-based EMS algorithm [4]. For computing iterative BP decoding thresholds, a q -ary extrinsic information transfer (EXIT) analysis was proposed in [5] and was later developed into a version suitable for *protograph-based* code ensembles in [6].

A protograph [7] is a small Tanner graph, which can be used to produce a *structured* LDPC code ensemble by applying a graph lifting procedure [8] with *lifting factor* M , such that every code in the ensemble is M times larger and maintains the structure of the protograph, i.e., it has the same degree distribution and the same type of edge connections. In this way, the computation graph [9] is maintained in the lifted graph [7], so BP threshold analysis can be performed on the protograph. A protograph consisting of $(c - b)$ check nodes and c variable nodes has *design rate* $R = b/c$ and can be represented equivalently by a $(c - b) \times c$ *base (parity-check) matrix* \mathbf{B} consisting of non-negative integers, in which the (i, j) -th entry ($1 \leq i \leq c - b$ and $1 \leq j \leq c$) is the number of edges connecting check node i and variable node j . Fig. 1 illustrates a $(3, 6)$ -regular protograph and its corresponding base matrix, which can be used to represent a $(3, 6)$ -regular LDPC-BC ensemble. To calculate the BP threshold of a protograph-based code ensemble, conventional tools are adapted to take the edge connections into account [7], [10]. Although some freedom is lost in the code design when the protograph structure is adopted,

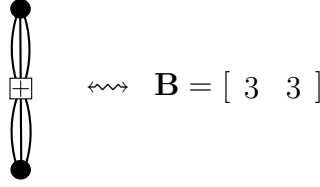


Fig. 1. A $(3, 6)$ -regular protograph and its corresponding base-matrix representation. Black circles correspond to variable nodes and crossed boxes correspond to check nodes.

one can use these modified protograph-based analysis tools to find “good” protograph-based ensembles with better BP thresholds than corresponding unstructured ensembles with the same degree distribution [10], [11].

Spatially coupled LDPC (SC-LDPC) codes, also known as terminated LDPC convolutional codes [12], are constructed by coupling together a series of L disjoint, or uncoupled, LDPC-BC Tanner graphs. Binary SC-LDPC code ensembles have been shown to exhibit a phenomenon called “threshold saturation” [13], [14], [15], in which, as the coupling length L grows, the BP decoding threshold saturates to the maximum *a-posteriori* (MAP) threshold of the corresponding uncoupled LDPC-BC ensemble, which, for the (J, K) -regular code ensembles considered in this paper, approaches channel capacity as the density of the parity-check matrix increases [16]. This threshold saturation phenomenon has been reported for a variety of code ensembles (e.g., (J, K) -regular SC-LDPC code ensembles [17], accumulate-repeat-by-4-jagged-accumulate (AR4JA) irregular SC-LDPC code ensembles [18], bilayer SC-LDPC code ensembles [19], and MacKay-Neal and Hsu-Anastasopoulos spatially-coupled code ensembles [20]) and channel models (e.g., channels with memory [21], multiple access channels [22], intersymbol-interference channels [23], and erasure relay channels [24]), thus making SC-LDPC codes attractive candidates for practical applications requiring near-capacity performance. For a more comprehensive survey of the literature on SC-LDPC codes, refer to the introduction of [25].

BP decoding threshold results on the BEC for q -ary SC-LDPC code ensembles have been reported by Uchikawa *et al.* [26] and Piemontese *et al.* [27], and the corresponding threshold saturation was proved by Andriyanova *et al.* [28]. In each of these papers, the authors assumed that decoding was simultaneously carried out across the entire parity-check matrix of the code; for simplicity, this will be referred to as flooding schedule decoding (FSD) in this paper. Employing

FSD for SC-LDPC codes can result in large latency, since a large coupling length L is needed to achieve near-capacity thresholds [25]. To resolve this issue, a more efficient technique, called windowed decoding (WD), was proposed in [29], [30] for binary SC-LDPC codes. Compared to FSD, WD exploits the convolutional nature of the SC parity-check matrix to localize decoding and thereby reduce latency. Under WD, the decoding window contains only a small portion of the parity-check matrix, and within that window, BP decoding is performed.

In this paper, assuming that the binary image of a codeword is transmitted, we analyze the WD thresholds of a variety of (J, K) -regular protograph-based q -ary SC-LDPC code ensembles constructed from the corresponding uncoupled q -ary (J, K) -regular LDPC-BC ensembles via the *edge-spreading* procedure [17], [25], where the finite field size is $q = 2^m$ and m is a positive integer. In particular,

- 1) For the BEC, we extend the q -ary density evolution (DE) analysis proposed in [31] to a protograph version and apply this analysis in conjunction with WD to obtain *windowed decoding thresholds* for q -ary SC-LDPC code ensembles;
- 2) For the binary-input additive white Gaussian noise channel (BIAWGNC) with binary phase-shift keying (BPSK) modulation, we obtain windowed decoding thresholds for q -ary SC-LDPC code ensembles by applying a protograph-based EXIT analysis (originally proposed for q -ary LDPC-BC ensembles [6]) in conjunction with WD.

In both cases, our primary contribution is to determine how much the decoding latency of WD can be reduced without suffering a loss in threshold. We observe that

- 1) Compared to FSD of the corresponding uncoupled q -ary LDPC-BC ensembles, WD of q -ary SC-LDPC code ensembles provides a threshold gain. This gain increases as the finite field size increases.
- 2) Compared to FSD of a given q -ary SC-LDPC code ensemble, WD provides significant reductions in both decoding latency and decoding complexity, and these reductions increase as the finite field size increases.
- 3) By carefully designing the protograph structure, using what we call a “type 2” edge-spreading format, WD provides near-capacity thresholds for q -ary SC-LDPC code ensembles, even when both the finite field size and the window size are relatively small.
- 4) When there is a constraint on decoding latency and operation close to the threshold of a

binary SC-LDPC code ensemble is required, using the non-binary counterpart can provide a significant reduction in decoding complexity.

The rest of the paper is organized as follows. Section II describes the construction of protograph-based q -ary SC-LDPC code ensembles and reviews the structure of WD. Then Sections III and IV present the WD thresholds of various q -ary SC-LDPC code ensembles for the BEC and the BIAWGNC, respectively, as the finite field size and/or the window size vary. The WD threshold is evaluated from two perspectives: first, as the window size increases, whether it achieves its best numerical value when the window size is small to moderate; second, as the finite field size increases, whether this achievable value approaches capacity. Also, the effects of different protograph constructions on the WD threshold are evaluated and discussed. Finally, Section V studies the decoding latency and complexity of q -ary SC-LDPC code ensembles and examines the latency, complexity, and performance tradeoffs of WD.

In summary, by examining various q -ary SC-LDPC code ensembles, we bring additional insight to three questions:

- 1) Why spatially coupled codes perform better than the corresponding uncoupled block codes,
- 2) Why windowed decoding is preferred to flooding schedule decoding, and
- 3) When non-binary codes should be used instead of binary codes.

The results of this paper provide theoretical guidance for designing and implementing practical q -ary spatially coupled LDPC codes suitable for windowed decoding [32].

II. WINDOWED DECODING OF PROTOGRAPH-BASED q -ARY SC-LDPC CODE ENSEMBLES

A. *Protograph-based q -ary SC-LDPC Code Ensembles*

A (J, K) -regular SC-LDPC code ensemble can be constructed from a (J, K) -regular LDPC-BC ensemble using the edge-spreading procedure [17], [25], described here in terms of protograph representations of the code ensembles. Take $J = 3$, $K = 6$ as an example. As shown in Fig. 2, instead of transmitting a sequence of codewords from the $(3, 6)$ -regular LDPC-BC ensemble independently at time instants $t = 1, 2, \dots, L$, edges from the variable nodes at time instant t , originally connected only to the check node at time instant t , are now “spread” to also connect to check nodes at time instants $t, t+1, \dots, t+w$; in this way, memory is introduced and the different time instants are “coupled” together, i.e., a terminated convolutional, or spatially coupled, coding

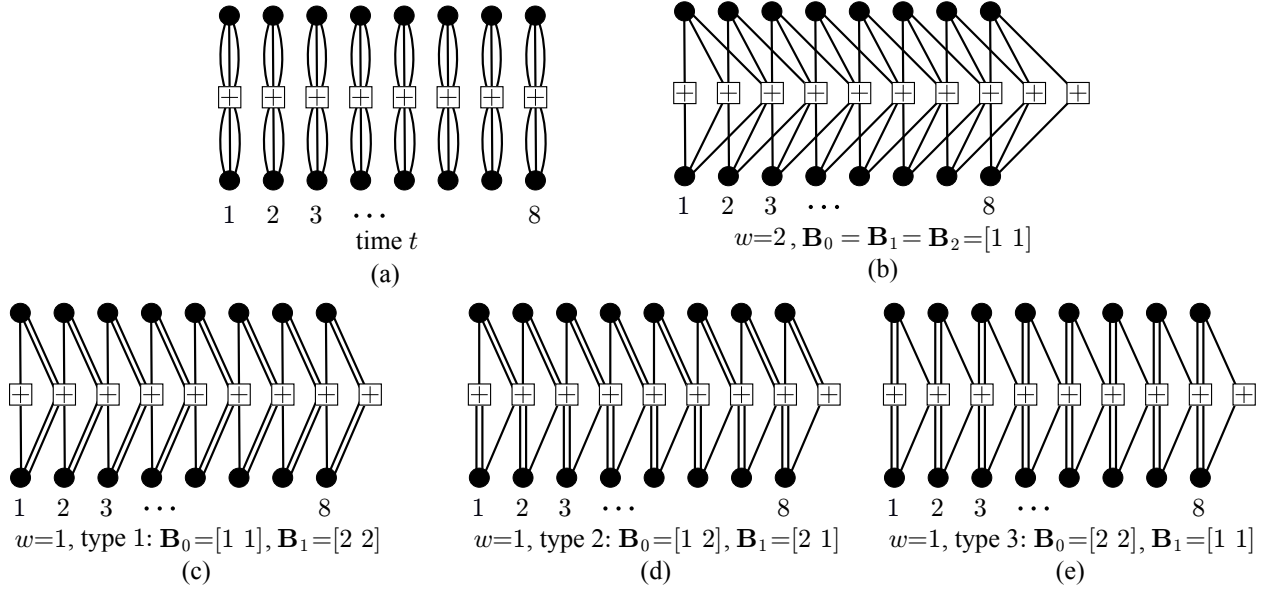


Fig. 2. (a) A sequence of $L = 8$ uncoupled $(3, 6)$ -regular LDPC-BC protographs, and (b)-(e) various $(3, 6)$ -regular SC-LDPC protographs constructed following the edge-spreading procedure with coupling length $L = 8$.

structure is introduced. The parameter w is referred to as the *coupling width*, and L is called the *coupling length*. Fig. 2 shows three different types of edge-spreading formats for $w = 1$ and one type for $w = 2$, all for the case $J = 3$, $K = 6$, and $L = 8$.

The above edge-spreading procedure can be described in terms of the base (parity-check) matrix representation of protographs as well. Let \mathbf{B} be a $(c-b) \times c$ block base matrix representing an LDPC-BC ensemble with design rate $R = b/c$. Then the base matrix of an SC-LDPC code ensemble can be constructed from \mathbf{B} as follows. First, \mathbf{B} is “spread” into a set of $(w + 1)$ *component base matrices* following the rule

$$\sum_{i=0}^w \mathbf{B}_i = \mathbf{B}, \quad (1)$$

so that each \mathbf{B}_i has the same size as \mathbf{B} . Next, an SC base matrix \mathbf{B}_{SC} is generated by “stacking and shifting” the base component matrices $\{\mathbf{B}_i\}_{i=0}^w$ at each time instant $t = 1, 2, \dots, L$, thereby

forming a convolutional structure:

$$\mathbf{B}_{\text{SC}} = \begin{bmatrix} \mathbf{B}_0 & & & & \\ \mathbf{B}_1 & \mathbf{B}_0 & & & \\ \vdots & \mathbf{B}_1 & \ddots & & \\ \mathbf{B}_w & \vdots & \ddots & \mathbf{B}_0 & \\ & \mathbf{B}_w & & \mathbf{B}_1 & \\ & & \ddots & \vdots & \\ & & & \mathbf{B}_w & \end{bmatrix}_{(L+w)(c-b) \times Lc}, \quad (2)$$

where the design rate of \mathbf{B}_{SC} is

$$R_L = 1 - \frac{(L+w)(c-b)}{Lc} = \frac{Lb - w(c-b)}{Lc}. \quad (3)$$

Due to the termination of \mathbf{B}_{SC} after Lc columns, there is a loss in the SC-LDPC code ensemble design rate R_L compared to the rate $R = b/c$ of \mathbf{B} . However, this rate loss diminishes as L increases and vanishes as $L \rightarrow \infty$, i.e., $\lim_{L \rightarrow \infty} R_L = R = b/c$.

Next, a finite-length q -ary SC-LDPC code is constructed from $\mathbf{B}_{\text{SC}} = [b_{i,j}]$ by following the procedure for constructing a finite-length q -ary LDPC-BC from \mathbf{B} :

- 1) “Lifting” [7]: Replace the nonzero entries $b_{i,j}$ in \mathbf{B}_{SC} with an $M \times M$ permutation matrix (or a sum of $b_{i,j}$ non-overlapping $M \times M$ permutation matrices if $b_{i,j} > 1$), and replace the zero entries with the $M \times M$ all-zero matrix, where M is called the lifting factor.
- 2) “Labeling”: Randomly assign to each non-zero entry in the lifted parity-check matrix a non-zero element uniformly selected from $\text{GF}(q)$, where $q = 2^m$ is the finite field size.

After the lifting step, the parity-check matrix is still binary, i.e., the non-binary feature does not arise until the labeling step.¹ The total code length is $n = LcM$, and we define the *constraint length* as the maximum width of the non-zero portion of the parity-check matrix $\nu = (w+1)cM$. Both the permutation matrices and the q -ary labels can be carefully chosen to obtain good codes with desirable properties. But constructing specific codes is not the emphasis of this paper; rather, we are interested in a threshold analysis of general q -ary ensembles consisting of all possible combinations of liftings and labelings of a given protograph, where the dimension of the message model used in the analysis depends on the size of the finite field [5], [31].

¹Note that “labeling” can come before “lifting”, resulting in a “constrained” protograph-based q -ary code as defined in [6].

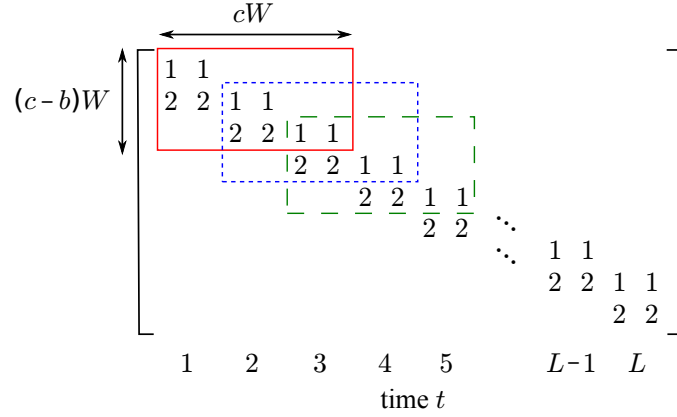


Fig. 3. WD example with window size $W = 3$: at $t = 1$ (solid red), $t = 2$ (dotted blue), and $t = 3$ (dashed green). $J = 3$, $K = 6$, $w = 1$; $\mathbf{B}_0 = [1, 1]$ and $\mathbf{B}_1 = [2, 2]$, both of size $(c-b) \times c = 1 \times 2$, for the \mathbf{B}_{SC} given by the protograph construction of Fig. 2(c). For each window position/time instant, the first $c = 2$ column blocks are target symbols.

B. Windowed Decoding (WD)

In this subsection, we briefly review the structure of WD. By construction, any two variable nodes (columns of the parity-check matrix) in the graph of an SC-LDPC code cannot be connected to the same check node if they are more than a constraint length $\nu = (w+1)cM$ (of columns) apart. As previously mentioned, compared to FSD, where iterative decoding is carried out on the *entire* parity-check matrix, WD of SC-LDPC code ensembles takes advantage of the convolutional structure of the parity-check matrix and localizes the decoding process to a small portion of the matrix, i.e., the BP algorithm is carried out only for those checks and variables covered by a “window”. Consequently, WD is an efficient way to reduce the memory and latency requirements of SC-LDPC codes [29], [30]. The WD algorithm can be described as follows (see [29] for further details):

- In terms of the SC base matrix \mathbf{B}_{SC} , the window is of fixed size $(c-b)W \times cW$ (recall that the size of the component base matrices \mathbf{B}_i 's in \mathbf{B}_{SC} is $(c-b) \times c$ measured in symbols, and slides from time instant $t = 1$ to time instant $t = L$, where W , called the *window size*, is defined as the number of column blocks of size c in the window. An example of WD with $W = 3$ is illustrated in Fig. 3 for the SC-LDPC code ensemble whose protograph is shown in Figure 2(c).
- At each time instant/window position, the BP algorithm runs until a fixed number of

iterations has been performed or some stopping rule [29], [30], [32] is satisfied, after which the window shifts c column blocks and those c column block symbols shifted out of the window are decoded. The first c column blocks in a window are called the *target symbols*. We assume that all the variables and checks in a window are updated during each iteration and that, after the window shifts, the final messages from the previously decoded target symbols are passed to the new window.

- Clearly, the largest possible W is equal to $(L + w)$, in which case the whole parity check matrix is covered and makes WD equivalent to FSD, and the smallest possible W is $(w + 1)$, i.e., the window length (measured in variables) when decoding an SC-LDPC code must be at least one constraint length. We are interested in searching for q -ary SC-LDPC code ensembles for which a small window size W can provide WD with a good threshold, which implies that the coupling width w should be kept small. Indeed, our results for q -ary SC-LDPC codes together with those in the literature for binary SC-LDPC codes [29], [30] show that ensembles with $w = 1$ provide the best latency-constrained performance with WD.

C. Code Ensemble Construction

In this paper, we restrict our attention to (J, K) -regular LDPC code ensembles.

1) (J, K) -regular LDPC-BC ensembles: Let

$$\mathbf{B} = \begin{bmatrix} J & J & \cdots & J \end{bmatrix}_{1 \times k} \quad (4)$$

denote the block base matrix corresponding to the protograph representation of a (J, K) -regular LDPC-BC ensemble, where $K = kJ$, $k = 1, 2, \dots$, and the design rate of the code ensemble is $R = (k - 1) / k$. That is, in the remainder of the paper, we let $c - b = 1$ and $c = k$. We denote the (J, K) -regular LDPC-BC ensemble constructed over $\text{GF}(2^m)$ as $\mathcal{B}(J, K, m)$.

2) *Edge spreadings of \mathbf{B}* : Given a variable node degree J , for a particular coupling width w , define

$$E(J, w) = \left\{ \begin{bmatrix} J_0 & J_1 & \cdots & J_w \end{bmatrix}^\top \middle| \sum_{i=0}^w J_i = J, J_i \in \{1, 2, \dots, J - w\} \right\}, \quad (5)$$

i.e., $E(J, w)$ is the set of all possible column vectors of length $(w + 1)$ satisfying the constraint $\sum_{i=0}^w J_i = J$, where $J_i \in \{1, 2, \dots, J - w\}$. Moreover, define \mathbf{B}_0^w as

$$\mathbf{B}_0^w = \begin{bmatrix} \mathbf{B}_0 \\ \mathbf{B}_1 \\ \vdots \\ \mathbf{B}_w \end{bmatrix}_{(w+1) \times k}, \quad (6)$$

i.e., \mathbf{B}_0^w is the “stack” of all the component base matrices $\{\mathbf{B}_i\}_{i=0}^w$. Then an edge-spreading format can be generated by selecting k elements (with replacement) from $E(J, w)$ as the k columns of \mathbf{B}_0^w . Recall from Section II-B that our major interest lies in q -ary SC-LDPC code ensembles for which windowed decoding (WD) achieves good thresholds under tight latency constraints, i.e., for a small window size W , which implies that the coupling width w should be small. Therefore, we do not allow w to exceed $(J - 1)$, i.e., the block base matrix \mathbf{B} should be spread into at most J component base matrices \mathbf{B}_i . In other words, for $E(J, w)$ in (5), we consider only values of w in the range $1 \leq w \leq J - 1$.

The edge-spreading format \mathbf{B}_0^w determines the SC base matrix \mathbf{B}_{SC} , and the q -ary WD thresholds depend on \mathbf{B}_{SC} . For a given \mathbf{B}_0^w , column permutations do not affect the WD threshold, but row permutations do. Consequently, for each combination of J and w , there will be $|E(J, w)| \cdot (1 + |E(J, w)|) / 2$ possible edge-spreading formats that can result in different WD thresholds. For example, consider the $(4, 8)$ -regular degree distribution with $J = 4$ and $w = 2$. Then

$$E(4, 2) = \left\{ \begin{bmatrix} 1 & 1 & 2 \end{bmatrix}^T, \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}^T, \begin{bmatrix} 2 & 1 & 1 \end{bmatrix}^T \right\}, \quad (7)$$

and the $|E(4, 2)| \cdot (1 + |E(4, 2)|) / 2 = 6$ possible edge-spreading formats that can give different WD thresholds are given by

$$\mathbf{B}_0^w \in \left\{ \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 2 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 2 \\ 1 & 1 \\ 2 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 2 & 2 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \right\}. \quad (8)$$

3) (J, K) -regular SC-LDPC code ensembles: We now detail the particular constructions of SC-LDPC code ensembles considered in the remainder of the paper. The first construction we

consider is the “classical” edge spreading [13] of the (J, K) -regular LDPC-BC base matrix \mathbf{B} given by (4), where $K = kJ$ and $w = J - 1$:

$$\mathbf{B}_0 = \mathbf{B}_1 = \cdots = \mathbf{B}_w = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}_{1 \times k}. \quad (9)$$

Unless noted otherwise, the coupling length for all the q -ary SC-LDPC code ensembles in this paper is taken to be $L = 100$, in order to keep the rate loss small. Consequently, we do not include L in the ensemble notation, and we denote as $\mathcal{C}_{J-1}(J, K, m)$ the SC-LDPC code ensemble constructed over $\text{GF}(2^m)$ using the component matrices $\mathbf{B}_0, \mathbf{B}_1, \dots, \mathbf{B}_w$ given by (9) in the base matrix \mathbf{B}_{SC} given by (2), with coupling width $w = J - 1$.

As noted previously, under tight latency constraints, the WD threshold can be improved by using small w ; in fact, excellent WD performance has been shown for binary SC-LDPC code ensembles using repeated edges in the protograph and $w = 1$ [29], [30]. In the case of q -ary SC-LDPC code ensembles, we have also found that the case $w = 1$, i.e., the set of edge spreadings

$$E(J, w = 1) = \left\{ \begin{bmatrix} 1 \\ J-1 \end{bmatrix}, \begin{bmatrix} 2 \\ J-2 \end{bmatrix}, \dots, \begin{bmatrix} J-1 \\ 1 \end{bmatrix} \right\}, \quad (10)$$

results in the best thresholds for low latency WD. Moreover, if we further restrict our attention to the edge-spreading pair

$$\mathbf{E}_A = \begin{bmatrix} 1 \\ J-1 \end{bmatrix}, \mathbf{E}_B = \begin{bmatrix} J-1 \\ 1 \end{bmatrix} \in E(J, 1), \quad (11)$$

we obtain the most interesting and representative constructions compared to the other possible selections of column vectors from $E(J, 1)$.

Combining \mathbf{E}_A and \mathbf{E}_B , there are $(k+1)$ possible choices for $\mathbf{B}_0^{w=1}$. An edge-spreading format is called “type- p ” if there are $(k-p+1)$ columns of \mathbf{E}_A in \mathbf{B}_0^1 followed by $(p-1)$ columns of \mathbf{E}_B , i.e.,

$$\begin{aligned} \mathbf{B}_0^1 &= \underbrace{\begin{bmatrix} \mathbf{E}_A & \cdots & \mathbf{E}_A \end{bmatrix}}_{k-p+1} \underbrace{\begin{bmatrix} \mathbf{E}_B & \cdots & \mathbf{E}_B \end{bmatrix}}_{p-1} \\ &= \underbrace{\begin{bmatrix} 1 & \cdots & 1 \\ J-1 & \cdots & J-1 \end{bmatrix}}_{k-p+1} \underbrace{\begin{bmatrix} J-1 & \cdots & J-1 \\ 1 & \cdots & 1 \end{bmatrix}}_{p-1} = \begin{bmatrix} \mathbf{B}_0 \\ \mathbf{B}_1 \end{bmatrix}, \end{aligned} \quad (12)$$

where $1 \leq p \leq k + 1$. Again, note that the ordering of columns is not important, because this simply results in column permutations of the resulting base matrix \mathbf{B}_{SC} and does not change the code or graph properties. We again omit L from the ensemble notation and denote as $\mathcal{C}_1(J, K, m, p)$ the type- p SC-LDPC code ensemble constructed over $\text{GF}(2^m)$ using component matrices \mathbf{B}_0 and \mathbf{B}_1 to form \mathbf{B}_{SC} , with coupling width $w = 1$, where $1 \leq p \leq k + 1$.

For a particular (J, K) pair and Galois field $\text{GF}(2^m)$, we refer informally to the collection of ensembles

$$\{\mathcal{B}(J, K, m), \mathcal{C}_{J-1}(J, K, m), \mathcal{C}_1(J, K, m, p) \mid p = 1, 2, \dots, k + 1\} \quad (13)$$

as “the (J, K, m) ensembles”, and we further refer to the collection of ensembles

$$\{\mathcal{C}_{J-1}(J, K, m), \mathcal{C}_1(J, K, m, p) \mid p = 1, 2, \dots, k + 1\} \quad (14)$$

as “the (J, K, m) SC ensembles”. For example, for an arbitrary m , let $(J, K) = (3, 6)$. In this case $k = 2$, and we consider the “classical” edge spreading with $w = J - 1 = 2$ along with $k + 1 = 3$ types of edge spreading with $w = 1$, viz.:

- $\mathcal{C}_2(3, 6, m)$: $\mathbf{B}_0 = \mathbf{B}_1 = \mathbf{B}_2 = \begin{bmatrix} 1 & 1 \end{bmatrix}$;
- $\mathcal{C}_1(3, 6, m, 1)$: $\mathbf{B}_0 = \begin{bmatrix} 1 & 1 \end{bmatrix}$, $\mathbf{B}_1 = \begin{bmatrix} 2 & 2 \end{bmatrix}$;
- $\mathcal{C}_1(3, 6, m, 2)$: $\mathbf{B}_0 = \begin{bmatrix} 1 & 2 \end{bmatrix}$, $\mathbf{B}_1 = \begin{bmatrix} 2 & 1 \end{bmatrix}$;
- $\mathcal{C}_1(3, 6, m, 3)$: $\mathbf{B}_0 = \begin{bmatrix} 2 & 2 \end{bmatrix}$, $\mathbf{B}_1 = \begin{bmatrix} 1 & 1 \end{bmatrix}$.

These four ensembles form the $(3, 6, m)$ SC ensembles, and together with $\mathcal{B}(3, 6, m)$ they form the $(3, 6, m)$ ensembles. Fig. 2 shows each of the $(3, 6, m)$ ensembles with coupling length $L = 8$ and arbitrary m .

III. THRESHOLD ANALYSIS OF q -ARY SC-LDPC CODE ENSEMBLES ON THE BEC

A. Protograph Density Evolution (DE) for q -ary LDPC Code Ensembles on the BEC

The q -ary DE algorithm presented in [31] was originally derived for randomized uncoupled q -ary LDPC-BC ensembles where 1) the symbol set is the vector space GF_2^m of dimension m over the binary field, and 2) the edge labeling set is the general linear group GL_2^m over the binary field, which is the set of all $m \times m$ invertible matrices whose entries are in $\{0, 1\}$. The thresholds of these code ensembles, as pointed out by the authors of [31], are very good approximations to those of q -ary LDPC-BC ensembles defined over $\text{GF}(2^m)$, since the numerical difference is on the order of 10^{-4} .

Consider an ordered list of the elements of GF_2^m , and assume that the zero element is in the 0th position of the list. For a specific code, a probability domain message vector in q -ary BP decoding is of length 2^m , where the entry at position i corresponds to the *a posteriori* probability that the symbol is the i -th element from GF_2^m . Since transmission is on the BEC and it can be assumed that the all-zero codeword is transmitted without affecting decoding performance [31], all the non-zero elements in the message vector must be equal; in fact, the set of symbols (elements from GF_2^m) whose *a posteriori* probabilities are non-zero forms a subspace of GF_2^m , and the message vector is said to have dimension n if it contains 2^n non-zero elements, $n = 0, 1, \dots, m$. Consequently, for the purpose of q -ary DE, which is concerned only with asymptotic ensemble-average properties rather than decoding a specific finite-length code, only the dimension of the BP decoding message vector needs to be tracked by the algorithm. As a result, a q -ary DE message vector for the BEC can be represented by a vector of length $(m+1)$, whose n -th entry, $n = 0, 1, \dots, m$, indicates the *a posteriori* probability that the BP decoding message vector has dimension n .

Similar to the procedure used to extend q -ary EXIT analysis to a protograph version in [6], we now extend the q -ary DE algorithm to a protograph version, which we refer to as q -ary protograph DE (PDE). Since the edge connections are taken into account and the computation graph is equal for all members of the ensemble, PDE reduces to the BP algorithm performed on the protograph. We use notation similar to that in [6] and [28]. Let $b_{i,j}$ denote a non-zero entry in the base matrix and recall that, from the perspective of the protograph, the value of $b_{i,j}$ is the number of edges connecting check node i (the row index in the matrix) to variable node j (the column index), rather than an edge label. Let $N(i)$ (resp. $M(j)$) denote the neighboring variables (resp. checks) of check i (resp. variable j). Let $\mathbf{p}_C^{(l)}(i, j)$ (resp. $\mathbf{p}_V^{(l)}(i, j)$) denote the check- i -to-variable- j (resp. variable- j -to-check- i) q -ary DE message vector during iteration l . Finally, let the erasure probability of the BEC be ϵ . Then the q -ary PDE algorithm consists of four steps as follows:

- Initialization: for each $b_{i,j} > 0$, let

$$\mathbf{p}_V^{(0)}(i, j) = \mathbf{p}_V^{(0)}(j) = \mathbf{p}^{(0)}(\epsilon), \quad (15)$$

where $\mathbf{p}^{(0)}(x)$ is a vector of length $(m+1)$ in the probability domain, whose n -th entry is

defined as

$$\binom{m}{n} x^n (1-x)^{m-n}. \quad (16)$$

- Check-to-variable update: the message vector from check i to variable j is

$$\mathbf{p}_C^{(l)}(i, j) = \left[\boxtimes_{s \in N(i) \setminus j} \left(\boxtimes^{b_{i,s}} \mathbf{p}_V^{(l-1)}(i, s) \right) \right] \boxtimes \left(\boxtimes^{b_{i,j}-1} \mathbf{p}_V^{(l-1)}(i, j) \right), \quad (17)$$

where the “ \boxtimes ” notation (see Appendix A of [28] for details) is described as follows. For two q -ary DE message vectors \mathbf{p}_1 and \mathbf{p}_2 , $\mathbf{p}_1 \boxtimes \mathbf{p}_2$ has n -th element

$$\sum_{i=0}^n \sum_{j=n-i}^n C_{i,j,n}^m p_{1,i} p_{2,j}, \quad (18)$$

where $p_{1,i}$ is the i -th element of \mathbf{p}_1 , $p_{2,j}$ is the j -th element of \mathbf{p}_2 ,

$$C_{i,j,n}^m = \frac{G_{m-i,m-n} G_{i,n-j} 2^{(n-i)(n-j)}}{G_{m,m-j}} \quad (19)$$

is the probability of choosing a subspace (of GF_2^m) of dimension j whose sum with a subspace of dimension i has dimension n , and

$$G_{m,k} = \begin{cases} 1 & \text{if } k = m \text{ or } k = 0, \\ \prod_{l=0}^{k-1} \frac{2^m - 2^l}{2^k - 2^l} & \text{if } 0 < k < m, \\ 0 & \text{otherwise,} \end{cases} \quad (20)$$

is the Gaussian binomial coefficient, the number of different subspaces of dimension k of GF_2^m . Finally, $\boxtimes^{b_{i,j}-1} \mathbf{p} = \mathbf{p} \boxtimes \mathbf{p} \boxtimes \dots \boxtimes \mathbf{p}$, with $(b_{i,j} - 1)$ occurrences of \mathbf{p} .

- Variable-to-check update: the message vector from variable j to check i is

$$\mathbf{p}_V^{(l)}(i, j) = \mathbf{p}_V^{(0)}(j) \boxdot \left[\boxdot_{s \in M(j) \setminus i} \left(\boxdot^{b_{s,j}} \mathbf{p}_C^{(l)}(s, j) \right) \right] \boxdot \left(\boxdot^{b_{s,j}-1} \mathbf{p}_C^{(l)}(s, j) \right), \quad (21)$$

where $\mathbf{p}_1 \boxdot \mathbf{p}_2$ has n -th element

$$\sum_{i=n}^m \sum_{j=n}^{m-i+n} V_{i,j,n}^m p_{1,i} p_{2,j}, \quad (22)$$

and

$$V_{i,j,n}^m = \frac{G_{i,n} G_{m-i,j-n} 2^{(i-n)(j-n)}}{G_{m,j}} \quad (23)$$

is the probability of choosing a subspace of dimension j whose intersection with a subspace of dimension i has dimension n (again, see Appendix A of [28] for details).

- Convergence check: the *a-posteriori* message vector for variable j is

$$\mathbf{p}_{\mathbf{V}, \text{APP}}^{(l)}(j) = \mathbf{p}_{\mathbf{V}}^{(0)}(j) \square \left[\square_{i \in M(j)} \left(\square^{b_{i,j}} \mathbf{p}_{\mathbf{C}}^{(l)}(i, j) \right) \right]. \quad (24)$$

The q -ary PDE algorithm ends when

- Either a decoding success is declared: for all the variables to be decoded, the 0th entry of each $\mathbf{p}_{\mathbf{V}, \text{APP}}^{(l)}(j)$ (denoted as $\mathbf{p}_{\mathbf{V}, \text{APP}}^{(l)}(j)[0]$) is at least $(1 - \delta)$, i.e., $\mathbf{p}_{\mathbf{V}, \text{APP}}^{(l)}(j)[0] \geq 1 - \delta$, where $\delta \in [0, 1]$ is a preset erasure rate,
- Or a decoding failure is declared: the algorithm reaches some maximum number of iterations.

The parameter δ should be chosen small enough so that it is essentially certain that q -ary PDE has converged if the condition is satisfied.

1) Flooding-Schedule Decoding (FSD) Thresholds for q -ary SC-LDPC Code Ensembles:

Given m characterizing the symbol set and ϵ characterizing the BEC, if q -ary PDE is performed over the entire base matrix \mathbf{B}_{SC} of an SC-LDPC code ensemble, then the algorithm determines asymptotically (i.e., for coupling length $L \rightarrow \infty$ and lifting factor $M \rightarrow \infty$) whether FSD can be successful on an ensemble average basis for that specific BEC. Thus, q -ary PDE can be used to calculate the FSD threshold, denoted $\epsilon^*(m, \delta)$, which is the largest channel erasure rate such that all transmitted symbols can be recovered successfully with probability at least $(1 - \delta)$, as the number of iterations l goes to infinity, i.e.,

$$\epsilon^*(m, \delta) = \sup \left\{ \epsilon \in [0, 1] \mid \mathbf{p}_{\mathbf{V}, \text{APP}}^{(l)}(j)[0] \geq 1 - \delta \text{ for } 1 \leq j \leq kL, \text{ as } l \rightarrow \infty \right\}. \quad (25)$$

The following numerical FSD threshold results on the BEC are obtained for $\delta = 10^{-6}$, and from this point forward $\epsilon^*(m, \delta)$ will be denoted simply as $\epsilon^*(m)$.

2) Windowed Decoding (WD) Thresholds for q -ary SC-LDPC Code Ensembles: We also apply q -ary PDE to WD in order to calculate the WD threshold of an SC-LDPC code ensemble defined over $\text{GF}(2^m)$.

The q -ary WD-PDE algorithm consists of performing q -ary PDE for all the window positions/time instants $t = 1, 2, \dots, L$, as illustrated in Fig. 3. For each window position, q -ary PDE is performed within the $W \times kW$ window; however, unlike the case of FSD, now the convergence check involves only the target symbols, i.e., the first k symbols in the window. Starting from $t = 1$, if q -ary PDE declares a decoding failure, then the whole q -ary WD-PDE terminates and

declares a decoding failure; otherwise, the window slides forward and q -ary PDE is performed for the next window position. The q -ary WD-PDE algorithm declares a decoding success for a specific BEC if and only if its “component” q -ary PDE declares decoding successes for all the window positions. Thus, given m , ϵ , and W , q -ary WD-PDE can be used to calculate the WD threshold of an SC-LDPC code ensemble.

We now define

$$\epsilon_{\text{WD}}^*(m, W, t, \delta) = \sup \left\{ \epsilon \in [0, 1] \left| \begin{array}{l} \mathbf{p}_{\text{V, APP}}^{(l)}(j)[0] \geq 1 - \delta \text{ for } tk - k + 1 \leq j \leq tk, \\ \text{as } l \rightarrow \infty \end{array} \right. \right\} \quad (26)$$

as the largest channel erasure rate such that all the target symbols in window position t can be recovered successfully with probability at least $(1 - \delta)$, as l goes to infinity, given that all the target symbols in the previous $(t - 1)$ windows have already been recovered successfully with probability at least $(1 - \delta)$. Then the WD threshold $\epsilon_{\text{WD}}^*(m, W, \delta)$ is the infimum of $\{\epsilon_{\text{WD}}^*(m, W, t, \delta)\}_{t=1}^L$, i.e.,

$$\epsilon_{\text{WD}}^*(m, W, \delta) = \inf_{1 \leq t \leq L} \epsilon_{\text{WD}}^*(m, W, t, \delta), \quad (27)$$

guaranteeing that all the transmitted symbols – consisting of all the target symbols in all the windows – can be recovered successfully with probability at least $(1 - \delta)$, as l goes to infinity.

It was proved in Proposition 1 of [29] that the WD thresholds of binary SC-LDPC code ensembles on the BEC are non-decreasing with increasing W , i.e., $\epsilon_{\text{WD}}^*(1, W, \delta) \leq \epsilon_{\text{WD}}^*(1, W + 1, \delta)$ for any $\delta \in [0, 1]$ and all W , $W = w + 1, w + 2, \dots, w + L$. By combining this proof with the monotonicity of q -ary variable and check node updates, proved in Appendix B of [28], we can state the following theorem.

Theorem 1 (Monotonicity of $\epsilon_{\text{WD}}^(m, W, \delta)$ with increasing W):* For a fixed $m \geq 1$, any $\delta \in [0, 1]$, and all W , $W = w + 1, w + 2, \dots, w + L$,

$$\epsilon_{\text{WD}}^*(m, W, \delta) \leq \epsilon_{\text{WD}}^*(m, W + 1, \delta). \quad (28)$$

As in the case of FSD thresholds, we choose $\delta = 10^{-6}$, and from this point forward $\epsilon_{\text{WD}}^*(m, W, \delta)$ will be denoted simply as $\epsilon_{\text{WD}}^*(m, W)$.

B. Numerical results: $k = 2$ ($R = 1/2$)

In this subsection we focus on the BP thresholds of the rate $R = 1/2$ q -ary SC-LDPC code ensembles with $k = 2$: in particular, we consider the $(2, 4)$ -, $(3, 6)$ -, $(4, 8)$ -, and $(5, 10)$ -regular

code ensembles. Our emphasis is on the scenario when WD is used, and the q -ary WD-PDE algorithm described in the previous subsection is adopted to calculate the corresponding BP thresholds.

Recall from Section II-C that, for $k = 2$, the SC-LDPC code ensembles we consider are the following:

$$\mathcal{C}_{J-1}(J, K, m) : \mathbf{B}_0 = \mathbf{B}_1 = \cdots = \mathbf{B}_{J-1} = \begin{bmatrix} 1 & 1 \end{bmatrix}; \quad (29)$$

$$\mathcal{C}_1(J, K, m, 1) : \mathbf{B}_0 = \begin{bmatrix} 1 & 1 \end{bmatrix}, \mathbf{B}_1 = \begin{bmatrix} J-1 & J-1 \end{bmatrix}; \quad (30)$$

$$\mathcal{C}_1(J, K, m, 2) : \mathbf{B}_0 = \begin{bmatrix} 1 & J-1 \end{bmatrix}, \mathbf{B}_1 = \begin{bmatrix} J-1 & 1 \end{bmatrix}; \quad (31)$$

$$\mathcal{C}_1(J, K, m, 3) : \mathbf{B}_0 = \begin{bmatrix} J-1 & J-1 \end{bmatrix}, \mathbf{B}_1 = \begin{bmatrix} 1 & 1 \end{bmatrix}. \quad (32)$$

The classical edge spreading results in the maximum coupling width $w = J - 1$ by choosing each \mathbf{B}_i in \mathbf{B}_0^w equal to $\begin{bmatrix} 1 & 1 \end{bmatrix}$. When $w = 1$, the type 1 and type 3 ensembles, $\mathcal{C}_1(J, K, m, 1)$ and $\mathcal{C}_1(J, K, m, 3)$, will have the same FSD threshold $\epsilon^*(m)$, since their SC base matrices are equal up to row permutations and the q -ary PDE algorithm is performed over the entire base matrix \mathbf{B}_{SC} . However, their WD thresholds are different. Type 2 has one column of \mathbf{B}_0^w that is the same as type 1 and the other column that is the same as type 3, so it is expected that its WD threshold will be between those of types 1 and 3.

1) The (2, 4) ensembles: When $(J, K) = (2, 4)$, all four types of edge spreading for q -ary SC-LDPC code ensembles are the same. For $m = 1, 2, \dots, 10$, the FSD and WD thresholds are shown in Fig. 4:

- Comparing $\mathcal{C}_1(2, 4, m)$ to $\mathcal{B}(2, 4, m)$, the improvement in the FSD threshold $\epsilon^*(m)$ introduced by the spatially coupled structure is negligible for small m . However, as m increases, $\epsilon^*(m)$ for $\mathcal{C}_1(2, 4, m)$ increases and approaches the BEC capacity of a rate $R = 1/2$ code ensemble.² This is consistent with the observations made in [26]. We note that the $\mathcal{B}(2, 4, m)$ ensembles do not display this behavior; in particular, their thresholds diverge from capacity as m increases, $m \geq 5$.
- For WD of $\mathcal{C}_1(2, 4, m)$ with fixed m , the threshold $\epsilon_{\text{WD}}^*(m, W)$ improves as the window size W increases – see Theorem 1 in Section III-A – and saturates numerically to a (maximum)

²Since $L = 100$, the design rate of $\mathcal{C}_1(2, 4, m)$ is $R_L = 0.495$ and capacity is $\epsilon_{\text{sh}} = 1 - 0.495 = 0.505$. This gap to capacity vanishes as $L \rightarrow \infty$, since the thresholds do not further decay and $R_L \rightarrow 1/2$.

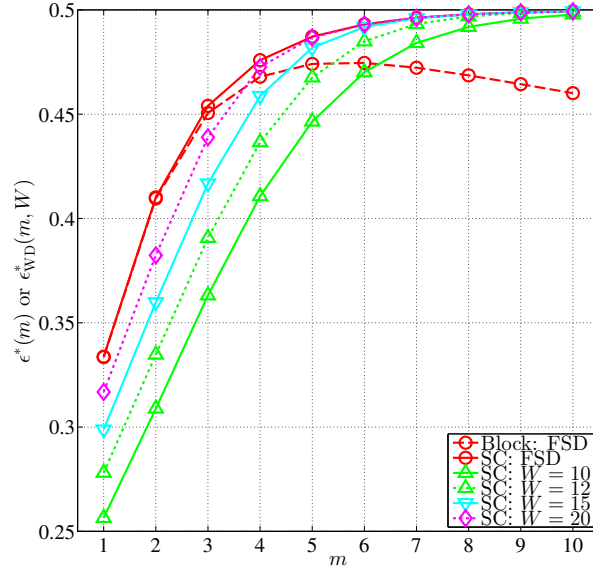


Fig. 4. FSD thresholds $\epsilon^*(m)$ and WD thresholds $\epsilon_{\text{WD}}^*(m, W)$ of $\mathcal{C}_1(2, 4, m)$.

constant value $\hat{\epsilon}_{\text{WD}}(m)$. Thus, we define

$$W^*(m) = \min \{W \mid \epsilon_{\text{WD}}^*(m, W) \cong \hat{\epsilon}_{\text{WD}}(m)\} \quad (33)$$

as the smallest window size that provides the best threshold $\hat{\epsilon}_{\text{WD}}(m)$ for a fixed m ; here, “ \cong ” is used to denote a numerically indistinguishable equality.³ We now make three observations regarding the ensemble $\mathcal{C}_1(2, 4, m)$:

- For all m , $\hat{\epsilon}_{\text{WD}}(m) = \epsilon^*(m)$, i.e., when the window size W is large enough, the WD threshold equals the FSD threshold.
- As m increases, $W^*(m)$ is non-increasing, i.e., increasing the finite field size can “speed up” the saturation of $\epsilon_{\text{WD}}^*(m, W)$ to $\hat{\epsilon}_{\text{WD}}(m)$ as W increases.
- The saturation of $\epsilon_{\text{WD}}^*(m, W)$ to $\hat{\epsilon}_{\text{WD}}(m)$ is relatively slow as W increases, especially when m is small. For example, when $m = 1$, we need a window size of $W^*(1) = 30$ to obtain the threshold $\hat{\epsilon}_{\text{WD}}(1)$. Moreover, even for a fairly large window, say $W = 20$, the WD threshold of $\mathcal{C}_1(2, 4, m)$ is worse than the FSD threshold of $\mathcal{B}(2, 4, m)$ for $m = 1, 2$, and 3 . This indicates that $\mathcal{C}_1(2, 4, m)$ does not perform well unless W and/or m are large.

³For our purposes, two thresholds are numerically indistinguishable if their absolute difference is no more than 10^{-6} .

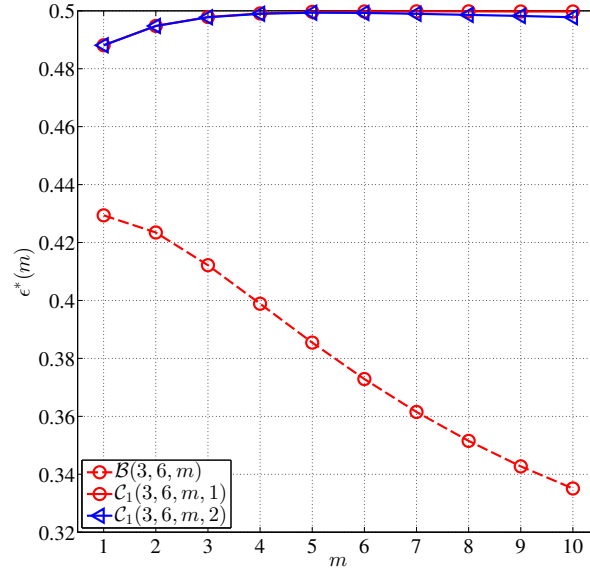


Fig. 5. FSD thresholds $\epsilon^*(m)$ comparison of the $(3, 6, m)$ ensembles.

As a result, we conclude that $\mathcal{C}_1(2, 4, m)$ is not a good candidate for WD, since a desirable q -ary SC-LDPC code ensemble should provide a near-capacity threshold when both the finite field size and the window size are relatively small, resulting in both small decoding latency and small decoding complexity – details will be discussed later in Section V. We will see in the remainder of this section, however, that increasing the node degrees in the code graph speeds up the saturation of $\epsilon_{\text{WD}}^*(m, W)$ to $\hat{\epsilon}_{\text{WD}}(m)$.

2) *The $(3, 6)$ ensembles:* As a benchmark, Fig. 5 compares the FSD thresholds of ensembles $\mathcal{C}_1(3, 6, m, 1)$ (and thus $\mathcal{C}_1(3, 6, m, 3)$) and $\mathcal{C}_1(3, 6, m, 2)$ to that of $\mathcal{B}(3, 6, m)$ for various m .⁴ It is observed that:

- For all finite field sizes 2^m , the introduction of the spatially coupled structure provides all four q -ary SC-LDPC code ensembles with significant improvement in the FSD threshold compared to the corresponding q -ary LDPC-BC ensemble. In fact, the gap between $\mathcal{B}(3, 6, m)$ and the $(3, 6, m)$ SC ensembles increases as m increases. Again, this is consistent with the observations made in [26] and [27].
- Note that, like the $\mathcal{B}(2, 4, m)$ ensembles discussed above, the $\mathcal{B}(3, 6, m)$ thresholds diverge

⁴Code ensembles $\mathcal{C}_2(3, 6, m)$ are not included in Fig. 5 because their thresholds are almost indistinguishable (although slightly different) from those of $\mathcal{C}_1(3, 6, m, 1)$.

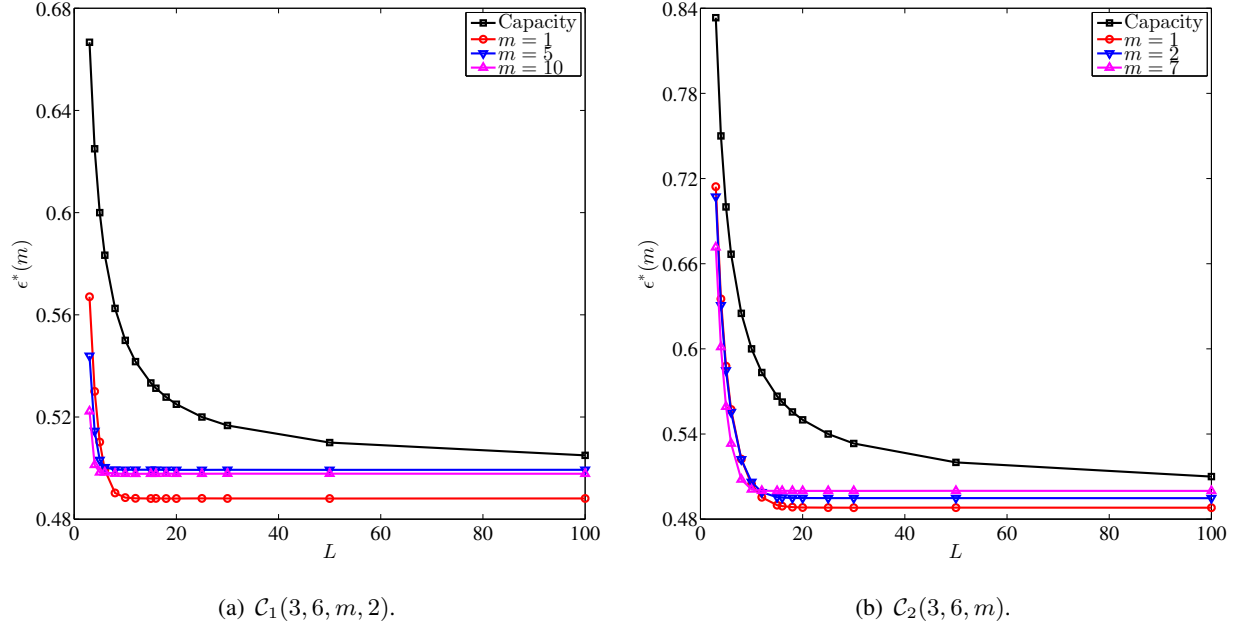


Fig. 6. FSD thresholds $\epsilon^*(m)$ of the SC-LDPC code ensembles with different coupling lengths L .

from capacity as m increases, while the FSD thresholds of $\mathcal{C}_1(3, 6, m, 1)$ and $\mathcal{C}_2(3, 6, m)$ increase and approach the BEC capacity for rate $R = 1/2$ as m increases. Surprisingly, this is not the case for $\mathcal{C}_1(3, 6, m, 2)$, whose FSD threshold increases and approaches capacity for $m = 5$, but then decreases slowly and thus diverges slightly from capacity as m increases further. As a result, in Fig. 5, there exists a small gap between the thresholds of $\mathcal{C}_1(3, 6, m, 1)$ and $\mathcal{C}_1(3, 6, m, 2)$ for large m .

We now briefly demonstrate the FSD threshold behavior of SC-LDPC code ensembles for varying coupling lengths L . Fig. 6 shows the FSD thresholds $\epsilon^*(m)$ for ensembles $\mathcal{C}_1(3, 6, m, 2)$ and $\mathcal{C}_2(3, 6, m)$ with increasing L . For fixed m and increasing L , the FSD thresholds initially decrease and then saturate to a constant value for sufficiently large L , which is consistent with results for binary protograph-based SC-LDPC code ensembles [13], [25].

Note that Figure 6 also illustrates the point made above that the $\mathcal{C}_1(3, 6, m, 2)$ ensemble does not have monotonically increasing thresholds with m . Specifically, in Fig. 6(a), for $\mathcal{C}_1(3, 6, m, 2)$, we have $\epsilon^*(1) < \epsilon^*(5)$ but $\epsilon^*(10) < \epsilon^*(5)$, while in Fig. 6(b), for $\mathcal{C}_2(3, 6, m)$, $\epsilon^*(m)$ increases uniformly as m increases: this confirms our observation of the small gap between the FSD thresholds of $\mathcal{C}_1(3, 6, m, 1)$ (almost indistinguishable from $\mathcal{C}_2(3, 6, m)$) and $\mathcal{C}_1(3, 6, m, 2)$ for

large m noted in Fig. 5.

With reference to Fig 6, given m , let $L^*(m)$ be the minimum L such that the threshold has saturated to its constant value, i.e.,

$$L^*(m) = \min \{L \mid \epsilon^*(m, L) \cong \epsilon^*(m, L'), L' = L + 1, L + 2, \dots\}. \quad (34)$$

As shown in Figs. 6(a) and 6(b), $L^*(m)$ is non-increasing as m increases; for example, for $\mathcal{C}_1(3, 6, m, 2)$, $L^*(1) = 15$, $L^*(3) = 10$, and $L^*(m) = 8$ when $m \geq 6$. Thus, we see that increasing the finite field size speeds up the saturation of the FSD threshold as L increases. To avoid repetition, we omit the FSD thresholds obtained for other (J, K) -regular SC-LDPC code ensembles with varying L ; however, it should be noted that the threshold saturation behavior described above is consistent over all considered code ensembles.

We now consider the WD thresholds of the $(3, 6, m)$ SC-LDPC code ensembles, again with $L = 100$. The WD thresholds of $\mathcal{C}_2(3, 6, m)$ with the classical edge-spreading format are shown in Fig. 7(a). As expected, for fixed m , the WD thresholds improve with increasing W , and we find that $\hat{\epsilon}_{\text{WD}}(m) = \epsilon^*(m)$ for $W \geq W^*(m)$, i.e., for a sufficiently large window, the WD threshold is equal to the FSD threshold for all m . We note that $W^*(m)$ is non-increasing as m increases, i.e., the saturation of the WD thresholds $\epsilon_{\text{WD}}^*(m, W)$ to $\hat{\epsilon}_{\text{WD}}(m)$ is faster for larger m . For example, $W^*(2) = 15$, $W^*(4) = 12$, and for $m \geq 7$, $W^*(m) = 8$. Due to a combination of the existence of degree-1 variable nodes in the window and the larger coupling width $w = 2$, $\mathcal{C}_2(3, 6, m)$ does not perform well using WD with a relatively small window.

Next, we consider the cases when $w = 1$: $\mathcal{C}_1(3, 6, m, 1)$, $\mathcal{C}_1(3, 6, m, 2)$, and $\mathcal{C}_1(3, 6, m, 3)$, shown in Figs. 7(b), 7(c), and 7(d), respectively. We observe that

- Similar to the $\mathcal{C}_2(3, 6, m)$ ensemble, for each of the three ensembles, at a particular m , the WD threshold $\epsilon_{\text{WD}}^*(m, W)$ improves as W increases and saturates numerically to a constant value $\hat{\epsilon}_{\text{WD}}(m)$. Again, increasing the finite field size speeds up the saturation as W increases; for example, $W^*(2) = 10$, $W^*(4) = 8$, and $W^*(6) = 6$ for $\mathcal{C}_1(3, 6, m, 1)$.
- Simply choosing $W \geq W^*(m)$ does not necessarily guarantee good WD thresholds, since $\hat{\epsilon}_{\text{WD}}(m)$ may not equal $\epsilon^*(m)$ even when W is large.⁵ In fact, $\hat{\epsilon}_{\text{WD}}(m) = \epsilon^*(m)$ for all

⁵Of course, as mentioned earlier, by selecting $W = L + w$ in WD, the decoding window covers the whole parity-check matrix and WD is equivalent to FSD. However, we are not considering this extreme case here.

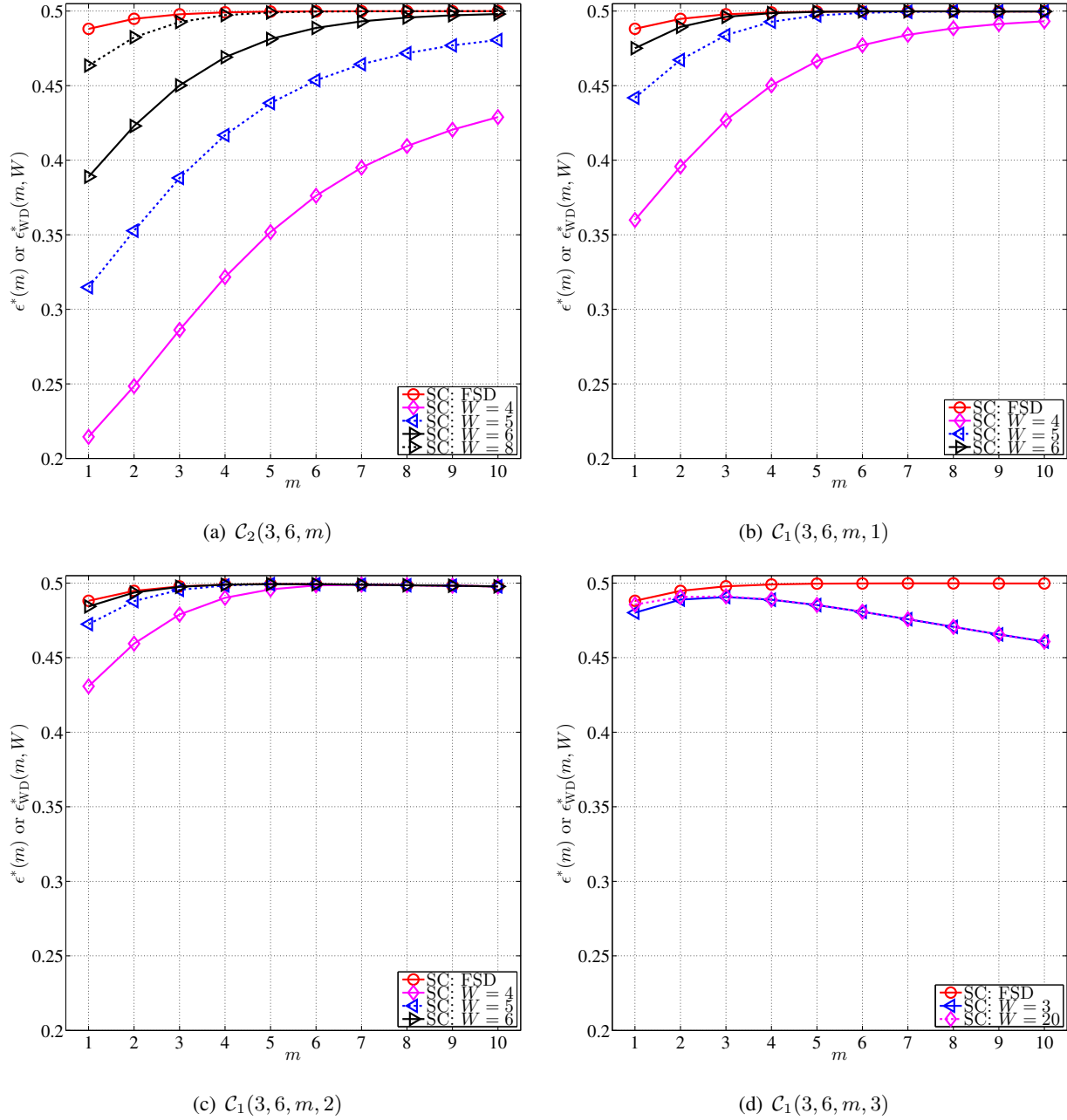


Fig. 7. WD thresholds $\epsilon_{\text{WD}}^*(m, W)$ of the $(3, 6, m)$ SC-LDPC code ensembles. FSD thresholds $\epsilon^*(m)$ are included as benchmarks.

m only for $C_1(3, 6, m, 1)$ and $C_1(3, 6, m, 2)$; for $C_1(3, 6, m, 3)$, on the other hand, $\hat{\epsilon}_{\text{WD}}(m)$ diverges from $\epsilon^*(m)$ as m increases, as shown in Fig. 7(d).

We turn our attention now to the implications of the WD thresholds on protograph design. Recall the three types of edge-spreading formats of the $(3, 6, m)$ SC ensembles with $w = 1$

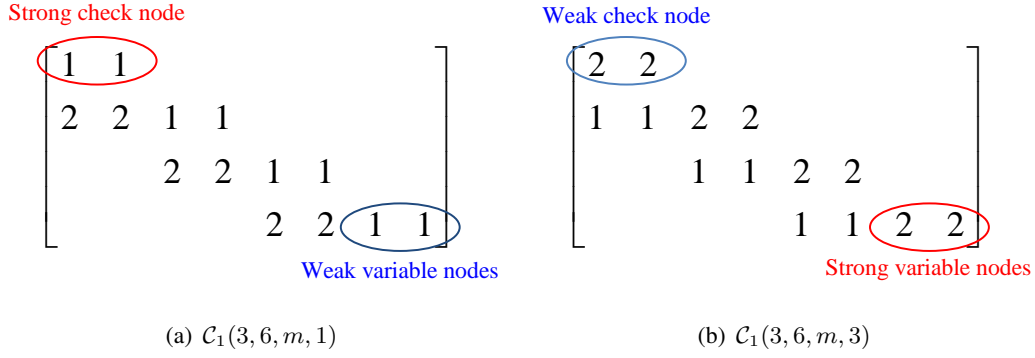


Fig. 8. The portion of the base matrix covered by the window when $W = 4$.

defined in (30)-(32), where \mathbf{B}_0^1 is given as $\begin{bmatrix} \mathbf{E}_A & \mathbf{E}_A \end{bmatrix}$, $\begin{bmatrix} \mathbf{E}_A & \mathbf{E}_B \end{bmatrix}$, and $\begin{bmatrix} \mathbf{E}_B & \mathbf{E}_B \end{bmatrix}$, respectively. As we move from type 1 to type 2 to type 3, the q -ary SC-LDPC code ensemble includes more $\mathbf{E}_B = \begin{bmatrix} 2 & 1 \end{bmatrix}^\top$ spreading and less $\mathbf{E}_A = \begin{bmatrix} 1 & 2 \end{bmatrix}^\top$ spreading. As illustrated in Fig. 8(a) for $\mathcal{C}_1(3, 6, m, 1)$ with a window size $W = 4$, \mathbf{E}_A spreading has a strong (lower degree) check node at the beginning of the window and weak variable nodes (with degree 1) at the end of the window. As a result, for all m , $\hat{\epsilon}_{\text{WD}}(m) = \epsilon^*(m)$ when W is large enough, but $\epsilon_{\text{WD}}^*(m, W)$ is not very good when W is relatively small – for example, $W = 4$ in Fig. 7(b). (See also the threshold behavior of the $\mathcal{C}_2(3, 6, m)$ ensembles in Fig. 7(a) which have a similar structure but larger w .)

On the other hand, as illustrated in Fig. 8(b) for $\mathcal{C}_1(3, 6, m, 3)$, \mathbf{E}_B spreading provides strong (higher degree) variable nodes at the end of the window and a weak (higher degree) check node at the beginning of the window. As a result, compared to $\mathcal{C}_1(3, 6, m, 1)$ and $\mathcal{C}_1(3, 6, m, 2)$, $\mathcal{C}_1(3, 6, m, 3)$ has the smallest $W^*(m)$ when m is fixed, i.e., threshold saturation to $\hat{\epsilon}_{\text{WD}}(m)$ is fastest as W increases, but $\hat{\epsilon}_{\text{WD}}(m)$ itself does not converge to $\epsilon^*(m)$, resulting in unsatisfactory WD thresholds, especially when m is large. In fact, comparing Fig. 7(d) to Fig. 5, we observe that the WD threshold of $\mathcal{C}_1(3, 6, m, 3)$ becomes more “block-like” as m increases, i.e., the curve for the WD threshold of $\mathcal{C}_1(3, 6, m, 3)$ behaves similarly to the curve for the FSD threshold of $\mathcal{B}(3, 6, m)$ for $m \geq 4$. This “block-like” behavior occurs for type 3 spreading because the edges of the block protograph have not been sufficiently spread, i.e., only one edge from each variable node in a block protograph is spread to the adjacent block protograph.

We summarize the above observations for WD thresholds with respect to the advantages and disadvantages of \mathbf{E}_A and \mathbf{E}_B spreading based on their effects on the portion of the parity-check matrix covered by the window:

- 1) The advantage of \mathbf{E}_A : Due to the strong check node at the start of the window, for a sufficiently large window size, the WD threshold saturates to the FSD threshold, which in turn approaches the channel capacity as the finite field size increases.
- 2) The disadvantage of \mathbf{E}_A : Due to the weak variable nodes at the end of the window, WD does not perform well when the window size is relatively small, so for small finite field sizes, there are large gaps between the WD threshold and the FSD threshold.
- 3) The advantage of \mathbf{E}_B : Due to the strong variable nodes at the start of the window, for relatively small window sizes, the WD threshold quickly saturates to its best achievable value, even for relatively small finite field sizes.
- 4) The disadvantage of \mathbf{E}_B : Due to the weak check node at the end of the window, WD tends to provide more “block-like” behavior, so that as the finite field size increases, the WD threshold diverges from the FSD threshold of the q -ary SC-LDPC code ensemble and approaches the FSD threshold of the corresponding uncoupled q -ary LDPC-BC ensemble.

Based on the advantages and disadvantages of these two antipolar spreading formats, we can develop design rules that combine fast saturation and FSD-achieving thresholds by mixing \mathbf{E}_B spreading and \mathbf{E}_A spreading, resulting in the type 2 spreading $\mathcal{C}_1(3, 6, m, 2)$. For example, as shown in Fig. 7(c), we see that $\mathcal{C}_1(3, 6, m, 2)$ has good WD thresholds even when both m and W are relatively small, i.e., with $m = 5$ and $W = 5$, the best performance is already achieved and lies within 0.15% of channel capacity. These design rules are consistent with the design rules proposed in [29] for the binary case, but they are more general in the sense that the effect of non-binary code alphabets is included.

To summarize, given the $(3, 6)$ -regular degree distribution, to achieve near-capacity thresholds with small decoding latency and small decoding complexity (see Section V for further details), the q -ary SC-LDPC code ensemble $\mathcal{C}_1(3, 6, m, 2)$ is recommended due to its excellent thresholds when the window size W and the finite field size q are both relatively small.

3) *The $(4, 8)$ and $(5, 10)$ ensembles:* We now examine the WD thresholds of the $(4, 8)$ -regular q -ary SC-LDPC code ensembles with $w = 1$ and the $(5, 10)$ -regular q -ary SC-LDPC code ensembles with $w = 1$ to explore how the advantages and disadvantages of \mathbf{E}_A and \mathbf{E}_B

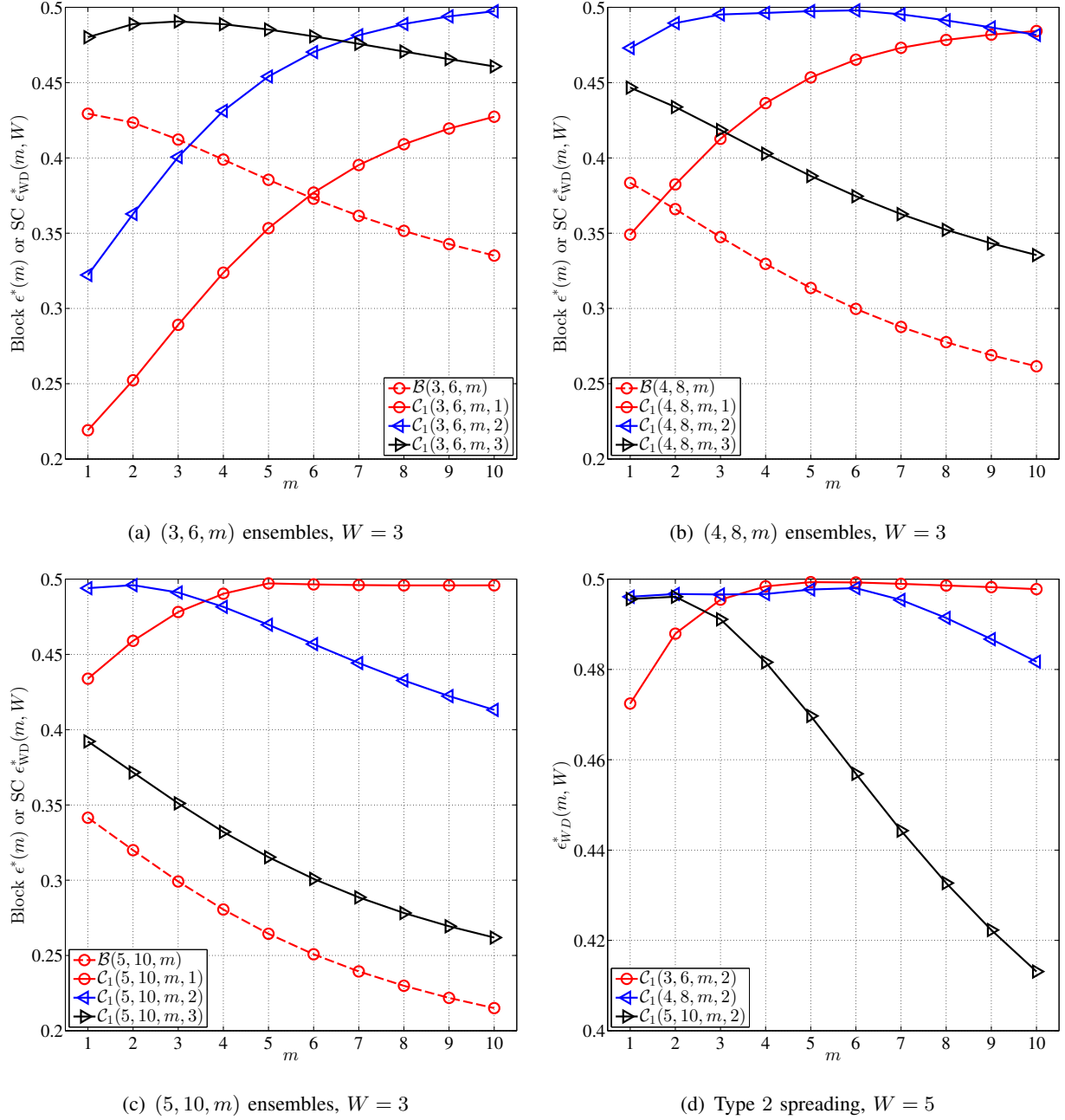


Fig. 9. WD thresholds $\epsilon_{\text{WD}}^*(m, W)$ of q -ary SC-LDPC code ensembles with $w = 1$ and $W = 3$: (a) the $(3, 6, m)$ ensembles, (b) the $(4, 8, m)$ ensembles, and (c) the $(5, 10, m)$ ensembles. FSD thresholds $\epsilon^*(m)$ of the corresponding q -ary LDPC-BC ensembles are included for reference. WD thresholds $\epsilon_{\text{WD}}^*(m, W)$ of the $C_1(J, 2J, m, 2)$ ensembles, $J = 3, 4$, and 5 , with $W = 5$ are shown in (d).

spreading are affected by the density (J, K) of the parity-check matrix, where we still have $k = K/J = 2$.

For comparison, the WD thresholds of the $(3, 6, m)$ SC ensembles with $w = 1$ and $W = 3$ are shown in Fig. 9(a), and the WD thresholds of the $(4, 8)$ and $(5, 10)$ SC ensembles with $w = 1$ and $W = 3$ are shown in Figs. 9(b) and 9(c), respectively. In addition to several features that are similar to the $(3, 6)$ SC ensembles, some further observations can be made for the $(4, 8)$ and $(5, 10)$ SC ensembles:

- Recall that for \mathbf{E}_B spreading, the advantage results from the strong variable nodes with degree $(J - 1)$ at the end of the window, and the disadvantage results from the weak check node with degree $2(J - 1)$ at the beginning of the window, as shown in Fig. 8(b) for $J = 3$. Thus, as the density J increases, both the positive and the negative effects are strengthened. On the one hand, the saturation of the WD threshold $\epsilon_{\text{WD}}^*(m, W)$ to its best achievable value $\hat{\epsilon}_{\text{WD}}(m)$ as W increases is faster. For example, for $m = 3$, we find that $W^*(3) = 4$ for $\mathcal{C}_1(3, 6, m, 3)$, $W^*(3) = 4$ for $\mathcal{C}_1(4, 8, m, 3)$, and $W^*(3) = 3$ for $\mathcal{C}_1(5, 10, m, 3)$, i.e., for fixed m , $W^*(m)$ is non-increasing for $\mathcal{C}_1(J, 2J, m, 3)$ as J increases. On the other hand, we observe from Fig. 9 that:
 - The WD thresholds of $\mathcal{C}_1(J, 2J, m, 3)$ monotonically decrease as m increases ($m \geq 3$ for $\mathcal{C}_1(3, 6, m, 3)$),
 - Their curves are almost parallel to the corresponding curves for the FSD thresholds of $\mathcal{B}(J, 2J, m)$ – this effect is more apparent for $J = 4$ and 5 , and
 - The gap between these two curves decreases as J increases.

Thus, the denser the parity-check matrix is, the more “block-like” the WD thresholds of type 3 spreading $\begin{bmatrix} \mathbf{E}_B & \mathbf{E}_B \end{bmatrix}$ become. As previously mentioned, this is because only one edge from each variable node in a block protograph is spread to the adjacent block protograph in type 3 edge spreading.

- The disadvantage of \mathbf{E}_B spreading also affects the WD thresholds of type 2 edge spreading. Fig. 9(d) compares the WD thresholds of the $\mathcal{C}_1(3, 6, m, 2)$, $\mathcal{C}_1(4, 8, m, 2)$, and $\mathcal{C}_1(5, 10, m, 2)$ ensembles with $W = 5$. We see that, as J increases, the thresholds of $\mathcal{C}_1(J, 2J, m, 2)$ diverge more significantly from channel capacity as m increases, consistent with the observation that the disadvantage of \mathbf{E}_B spreading is strengthened as J increases. Moreover, the divergence occurs sooner as J increases, e.g., the WD threshold of $\mathcal{C}_1(5, 10, m, 2)$ increases only up to $m = 2$ and then starts to decrease as m increases further, whereas the divergence for both

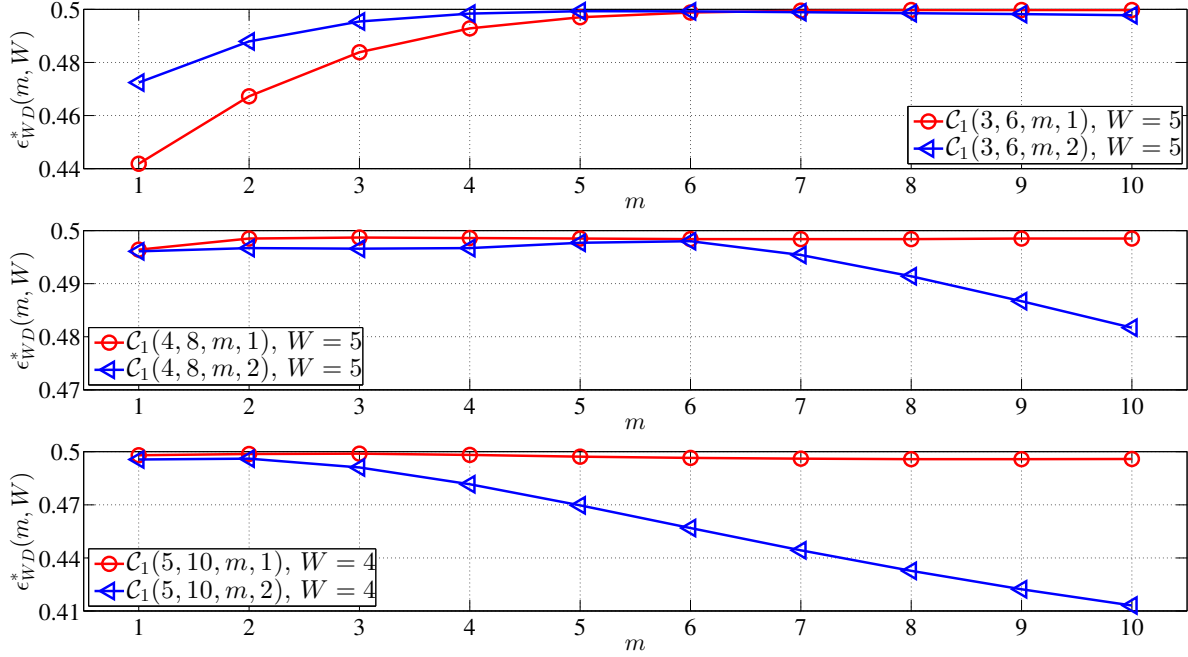


Fig. 10. Comparison of WD thresholds $\epsilon_{\text{WD}}^*(m, W)$: type-I spreading vs. type-II spreading for $J = 3$ with $W = 5$, for $J = 4$ with $W = 5$, and for $J = 5$ with $W = 4$. Note that for the latter two comparisons, $\mathcal{C}_1(J, 2J, m, 1)$ has better thresholds than $\mathcal{C}_1(J, 2J, m, 2)$ for all m .

$\mathcal{C}_1(3, 6, m, 2)$ and $\mathcal{C}_1(4, 8, m, 2)$ does not occur until $m = 6$.

- For type 1 edge spreading, where both columns of \mathbf{B}_0^1 equal \mathbf{E}_A , the WD thresholds improve dramatically as J increases for small W , as we see in Fig. 9 for $W = 3$. In other words, to a certain extent, the negative effect of \mathbf{E}_A spreading due to the presence of the degree-1 variable nodes at the end of the window, which results in poor WD thresholds for small W , is compensated for by the increased density of the parity-check matrix. This observation is further supported by Fig. 10, which compares the WD thresholds of $\mathcal{C}_1(J, 2J, m, 1)$ and $\mathcal{C}_1(J, 2J, m, 2)$ for $J = 3$ with $W = 5$, for $J = 4$ with $W = 5$, and for $J = 5$ with $W = 4$. We observe that for $J = 4$ and $J = 5$, $\mathcal{C}_1(J, 2J, m, 1)$ has better thresholds than $\mathcal{C}_1(J, 2J, m, 2)$ for all finite field sizes, even with relatively small W . This indicates that, although $\mathcal{C}_1(3, 6, m, 1)$ does not perform as well as $\mathcal{C}_1(3, 6, m, 2)$ with WD, $\mathcal{C}_1(4, 8, m, 1)$ and $\mathcal{C}_1(5, 10, m, 1)$ are both excellent choices for use with WD.

Based on the above observations, since the thresholds of type 2 spreading $\begin{bmatrix} \mathbf{E}_A & \mathbf{E}_B \end{bmatrix}$ deteriorate

rate as J increases (see Fig. 9(d)), while the thresholds of type 1 spreading $\begin{bmatrix} \mathbf{E}_A & \mathbf{E}_A \end{bmatrix}$ improve, we conclude for these two edge-spreading formats that

- 1) When $J = 3$, $\mathcal{C}_1(3, 6, m, 2)$ is better for WD than $\mathcal{C}_1(3, 6, m, 1)$,
- 2) When $J = 4$, both $\mathcal{C}_1(4, 8, m, 1)$ (for all m) and $\mathcal{C}_1(4, 8, m, 2)$ (for $m \leq 6$) give excellent performance with WD, and
- 3) When $J = 5$, $\mathcal{C}_1(5, 10, m, 1)$ is a better choice for WD than $\mathcal{C}_1(5, 10, m, 2)$.

Moreover, for $J = 3$, if the code construction is restricted to a very small finite field size – say $m = 1$ ($q = 2$) or $m = 2$ ($q = 4$) – and the threshold requirement can be slightly relaxed, then $\mathcal{C}_1(3, 6, m, 3)$ with $\begin{bmatrix} \mathbf{E}_B & \mathbf{E}_B \end{bmatrix}$ spreading also performs well for WD (see Fig. 7(d)). Again, this is consistent with the design rules proposed in [29] for binary SC-LDPC code ensembles suitable for WD. Finally, $\mathcal{C}_1(4, 8, m, 3)$ and $\mathcal{C}_1(5, 10, m, 3)$ ensembles are clearly not suitable for WD, as shown in Figs. 9(b) and 9(c).

The key point we wish to make throughout the paper is that desirable protograph-based q -ary SC-LDPC code ensembles for windowed decoding should achieve good thresholds when both the finite field size q and the window size W are small. To this end, we can summarize the above observations made for the $(3, 6)$, $(4, 8)$, and $(5, 10)$ SC ensembles with $w = 1$ into two design rules as follows:

- Combining \mathbf{E}_A spreading and \mathbf{E}_B spreading, i.e., type 2 edge spreading, is attractive when J , characterizing the density of the parity-check matrix, is small;
- As J increases, \mathbf{E}_B spreading becomes less attractive, and it should be totally avoided in favor of \mathbf{E}_A spreading when $J \geq 5$.

The classical $(4, 8)$ -regular and $(5, 10)$ -regular q -ary SC-LDPC code ensembles with $w = J-1$, i.e., $\mathcal{C}_3(4, 8, m)$ and $\mathcal{C}_4(5, 10, m)$ defined by (29), provide WD thresholds analogous to $\mathcal{C}_2(3, 6, m)$. To be more specific, for all m , the WD thresholds $\epsilon_{\text{WD}}^*(m, W)$ improve with W and saturate to $\hat{\epsilon}_{\text{WD}}(m) = \epsilon^*(m)$ (the FSD thresholds), which are non-decreasing and numerically achieve capacity as m increases. Further, when m is fixed and W is sufficiently large, the WD thresholds of the $\mathcal{C}_{J-1}(J, 2J, m)$ ensembles improve as J increases, as shown in Fig. 11 for $m = 2$, $W = 8$ and 10. Nevertheless, when W is small to moderate, the thresholds are not satisfactory; for example, when $W = 6$, there is still significant space for threshold improvement by increasing W further. In fact, since the minimum W required for WD of $\mathcal{C}_{J-1}(J, 2J, m)$ is $(w + 1) = J$,

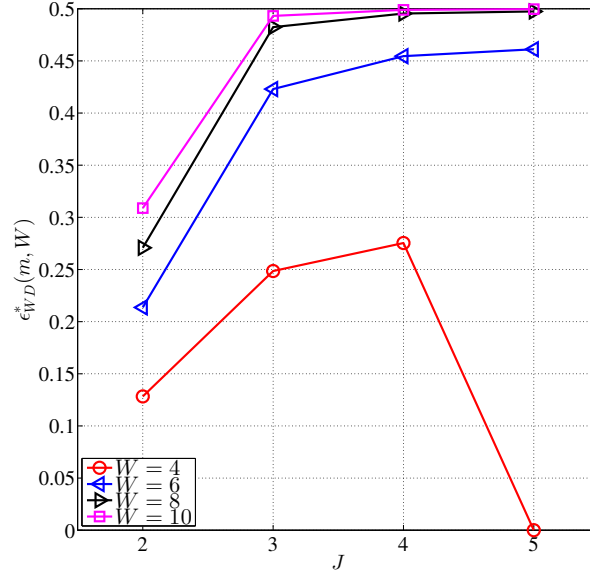


Fig. 11. WD thresholds $\epsilon_{WD}^*(m, W)$ of $\mathcal{C}_{J-1}(J, 2J, m=2)$ ensembles as J increases for window sizes $W = 4, 6, 8$, and 10 . (Note that for $\mathcal{C}_4(5, 10, m=2)$ with $W = 4$, $\epsilon_{WD}^*(m, W) = 0$, because the minimum required W is 5 .)

as J increases the classical edge spreading format is even less attractive if there is a constraint on decoding latency, i.e., if a small W must be adopted. For example, $\epsilon_{WD}^*(m, W=4) = 0$ for $\mathcal{C}_4(5, 10, m)$, as shown in Fig. 11 for $m=2$, because the minimum required window size in this case is $W=5$.

C. Numerical results: $k=3$ and $k=4$ ($R > 1/2$)

The previous subsection presented the advantages and disadvantages of using \mathbf{E}_A spreading and \mathbf{E}_B spreading in the construction of rate $R = 1/2$ protograph-based q -ary SC-LDPC code ensembles suitable for WD, and results were presented on the influence of varying the density J (and thus K) of the parity-check matrix on the WD thresholds. This subsection presents additional results on WD thresholds for higher rate ($R = 2/3$ and $3/4$) protograph-based q -ary SC-LDPC code ensembles, with emphasis on how the particular mix of \mathbf{E}_A spreading and \mathbf{E}_B spreading affects the WD thresholds. We expect that the more a certain kind of spreading is used, the more its corresponding advantages and disadvantages will be observed. For simplicity, we fix $J=3$.

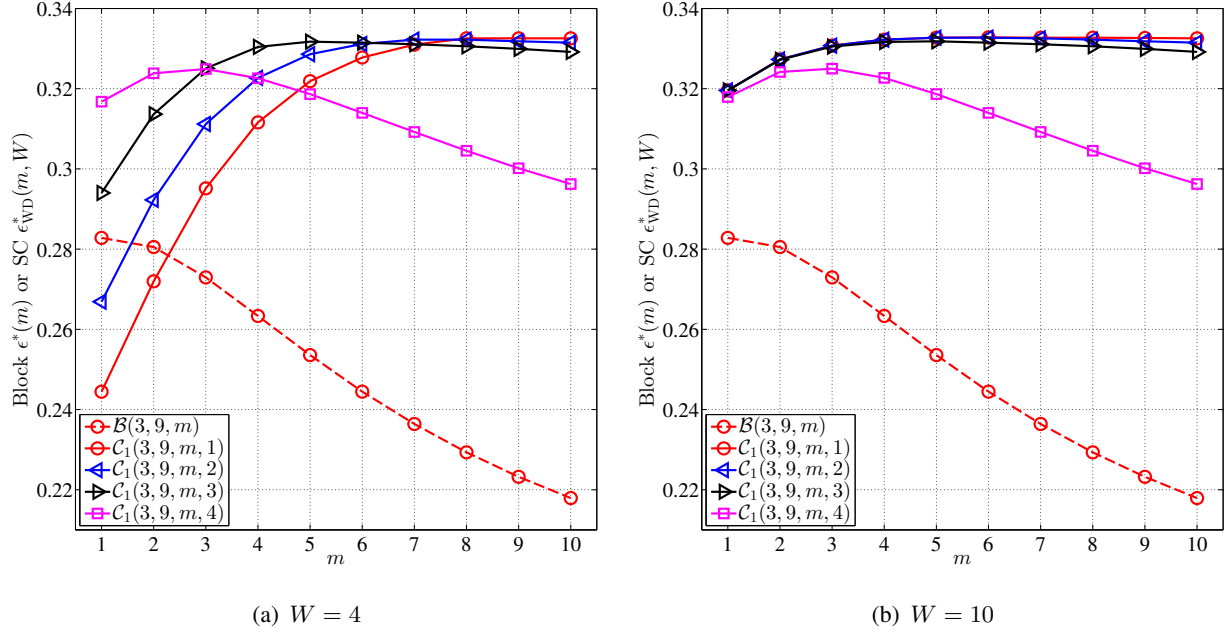


Fig. 12. WD thresholds $\epsilon_{\text{WD}}^*(m, W)$ of the $(3, 9, m)$ SC code ensembles with $w = 1$: (a) $W = 4$, and (b) $W = 10$, a sufficiently large window size such that the best WD thresholds are achieved for all the SC-LDPC code ensembles. FSD thresholds $\epsilon^*(m)$ of $\mathcal{B}(3, 9, m)$ are included as a benchmark.

1) $(J, K) = (3, 9)$, $k = 3$: We consider the $(3, 9, m)$ SC code ensembles over $\text{GF}(2^m)$ defined by (13). The asymptotic rate of $(3, 9)$ -regular q -ary SC-LDPC code ensembles is $R = (k-1)/k = 2/3$, when the coupling length L goes to infinity. Since $k = 3$, the component matrices \mathbf{B}_i used to construct \mathbf{B}_{SC} in (2) are of size 1×3 and, in addition to the classical edge spreading with $w = 2$, there are four types of $w = 1$ spreading where, for types 1 through 4, \mathbf{B}_0^1 is given as $\begin{bmatrix} \mathbf{E}_A & \mathbf{E}_A & \mathbf{E}_A \end{bmatrix}$, $\begin{bmatrix} \mathbf{E}_A & \mathbf{E}_A & \mathbf{E}_B \end{bmatrix}$, $\begin{bmatrix} \mathbf{E}_A & \mathbf{E}_B & \mathbf{E}_B \end{bmatrix}$, and $\begin{bmatrix} \mathbf{E}_B & \mathbf{E}_B & \mathbf{E}_B \end{bmatrix}$, respectively.

We expect that if there are more \mathbf{E}_B spreadings, then, for fixed m , the WD threshold $\epsilon_{\text{WD}}^*(m, W)$ will saturate faster to its best achievable value $\hat{\epsilon}_{\text{WD}}(m)$ as W increases, and that if there are more \mathbf{E}_A spreadings, then $\hat{\epsilon}_{\text{WD}}(m)$ will diverge less from channel capacity as m increases. These expectations are met, as illustrated in Figs. 12(a) and 12(b) when $W = 4$ and 10, respectively. Combined with other numerical results, it is observed that

- When m is fixed, the WD threshold $\epsilon_{\text{WD}}^*(m, W)$ of $\mathcal{C}_1(3, 9, m, 4)$, which contains all \mathbf{E}_B spreadings, has the fastest saturation to the corresponding $\hat{\epsilon}_{\text{WD}}(m)$ of all the $w = 1$ code ensembles. This indicates that there is little room for threshold improvement for

$\mathcal{C}_1(3, 9, m, 4)$ by increasing W to a large value; indeed, comparing the two $\mathcal{C}_1(3, 9, m, 4)$ curves in Figs. 12(a) and 12(b), we observe that, over the entire range of m , $\epsilon_{\text{WD}}^*(m, W)$ hardly changes when the window size goes from small ($W = 4$) to large ($W = 10$). In fact, even in the case $m = 1$ with the slowest saturation ($W^*(m) = 6$) among all field sizes, $\epsilon_{\text{WD}}^*(m, W = 4)$ for $\mathcal{C}_1(3, 9, m, 4)$ already lies within 0.35% of $\hat{\epsilon}_{\text{WD}}(m)$.

- On the other hand, this fast saturation of $\epsilon_{\text{WD}}^*(m, W)$ to $\hat{\epsilon}_{\text{WD}}(m)$ for type 4 edge spreading is accompanied by reduced threshold values. In Fig. 12(a), where W is small, for $m \leq 3$ the ordering of the ensemble types from best to worst is 4, 3, 2, 1, and WD of $\mathcal{C}_1(3, 9, m, 1)$ has even worse performance than FSD of the block code ensemble $\mathcal{B}(3, 9, m)$ for $m \leq 2$. When $W < W^*(m)$, the WD thresholds are worse than $\hat{\epsilon}_{\text{WD}}(m)$ because the decoder performance is impaired. In this regime, any additional structural weakness, such as weak variable nodes arising from an \mathbf{E}_A spreading, further harm the threshold, especially for small m , where we observe that fewer \mathbf{E}_A spreadings result in better thresholds. However, for a larger window size, the decoder is more robust, in the sense that some weaker variable nodes can be included without significantly harming performance. This allows for a stronger check node at the start of the window to initiate the “wave-like” decoding that results in threshold saturation for SC-LDPC code ensembles. This effect is more obvious in Fig. 12(b), where the window size $W = 10$ is chosen to be sufficiently large such that $\epsilon_{\text{WD}}^*(m, 10) = \hat{\epsilon}_{\text{WD}}(m)$, i.e., $W = 10 \geq W^*(m)$, for each SC code ensemble. Compared to Fig 12(a), for $m \leq 3$, types 1, 2, and 3 now provide almost identical WD thresholds, which are all better than type 4. In this regime, we clearly favor an edge-spreading format with a mixture of \mathbf{E}_A and \mathbf{E}_B .
- The introduction of \mathbf{E}_B spreadings causes a divergence from capacity $\epsilon_{\text{Sh}} = 1/3$ of a rate- $R = 2/3$ code ensemble as m increases. This is observed whether the window size is small (Fig. 12(a)) or large (Fig. 12(b)), and the divergence becomes more significant as more \mathbf{E}_B spreadings are used. This behavior is similar to what was observed for $\mathcal{C}_1(3, 6, m, 3)$ in Fig. 9(a), and again the “block-like” behavior as m increases can be explained by insufficient edge spreading to the adjacent block protograph.
- Finally, the $\mathcal{C}_1(3, 9, m, 1)$ ensembles, with all \mathbf{E}_A spreading, and the $\mathcal{C}_2(3, 9, m)$ classical edge spreading ensembles with $w = 2$ display non-decreasing maximum WD thresholds $\hat{\epsilon}_{\text{WD}}(m)$ that approach channel capacity as m increases. However, the weak variable nodes

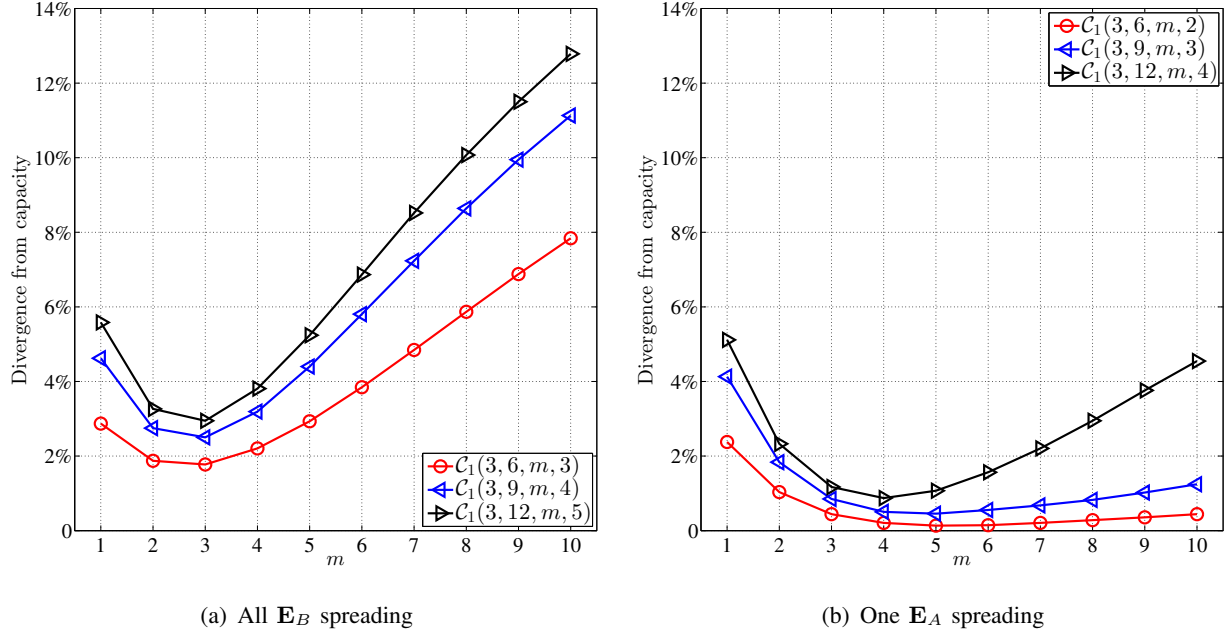


Fig. 13. Percentage divergence of the best achievable WD thresholds $\hat{\epsilon}_{\text{WD}}(m)$ from the corresponding channel capacities for (a) the $C_1(3, 6, m, 3)$, $C_1(3, 9, m, 4)$, and $C_1(3, 12, m, 5)$ ensembles with all \mathbf{E}_B spreading, and (b) the $C_1(3, 6, m, 2)$, $C_1(3, 9, m, 3)$, and $C_1(3, 12, m, 4)$ ensembles containing only one \mathbf{E}_A spreading.

at the end of the windows for these two ensembles imply that when m is small, $W^*(m)$ should be large.

2) $(J, K) = (3, 12)$, $k = 4$: The asymptotic rate of $(3, 12)$ -regular q -ary SC-LDPC code ensembles is $R = (k - 1)/k = 3/4$, when the coupling length L goes to infinity. For $w = 1$ and an arbitrary m , there are $k + 1 = 5$ types of $(3, 12, m)$ SC ensembles over $\text{GF}(2^m)$ defined in (13), and the behavior of their thresholds is similar to the $(3, 9, m)$ SC ensembles with $w = 1$. Fig. 13(a) shows the percentage divergence of the $\hat{\epsilon}_{\text{WD}}(m)$ from the corresponding channel capacities for the $C_1(3, 6, m, 3)$, $C_1(3, 9, m, 4)$, and $C_1(3, 12, m, 5)$ ensembles, where all \mathbf{E}_B spreading is adopted in each case. The results strengthen our observation that the more a particular spreading is used, the greater are its effects: the WD threshold of $C_1(3, 12, m, 5)$ shows the most significant divergence from the corresponding BEC capacity because it uses four \mathbf{E}_B spreadings, compared to three in $C_1(3, 9, m, 4)$ and two in $C_1(3, 6, m, 3)$.

Similar observations can be made in Fig. 13(b) as well, which compares the percentage divergence for the $C_1(3, 6, m, 2)$, $C_1(3, 9, m, 3)$, and $C_1(3, 12, m, 4)$ ensembles, where one \mathbf{E}_A

spreading and $(k - 1)$ \mathbf{E}_B spreadings are adopted in each case. Again, the ensemble that uses the most \mathbf{E}_B spreadings – in this case $\mathcal{C}_1(3, 12, m, 4)$ with three \mathbf{E}_B 's – shows the most significant divergence as m increases. However, compared to the thresholds in Fig. 13(a) with the same degree distribution ($J = 3, K = 3k$), we observe that introducing only one \mathbf{E}_A spreading can significantly alleviate the divergence effect of the \mathbf{E}_B spreading(s) and thus improve the WD thresholds, i.e., it is desirable to mix \mathbf{E}_A and \mathbf{E}_B spreadings in designing of WD-suitable code ensembles.

Finally, classical edge spreading of the $\mathcal{C}_{J-1}(J, 3J, m)$ and $\mathcal{C}_{J-1}(J, 4J, m)$ code ensembles with $w = J - 1$ is not suitable for WD, despite their excellent capacity-achieving thresholds when m and W are both large enough, as noted previously for the $\mathcal{C}_{J-1}(J, 2J, m)$ code ensembles.

We emphasize again the design rule that combining \mathbf{E}_B spreading and \mathbf{E}_A spreading is a good strategy for designing (J, K) -regular q -ary SC-LDPC code ensembles suitable for windowed decoding when J is small for two reasons:

- 1) The coupling width $w = 1$ makes the minimum required window size only $W = w + 1 = 2$, and
- 2) The threshold can be near capacity when m and W are both small.

The above conclusions are supported by WD threshold results for the $\mathcal{C}_1(3, 6, m, 2)$, $\mathcal{C}_1(4, 8, m, 2)$, $\mathcal{C}_1(3, 9, m, 2)$, $\mathcal{C}_1(3, 9, m, 3)$, $\mathcal{C}_1(3, 12, m, 2)$, and $\mathcal{C}_1(3, 12, m, 3)$ ensembles. For the cases when $J = 3$, these conclusions are further reinforced by decoding performance simulations of finite-length codes with different rates; see [32] for details.

IV. THRESHOLD ANALYSIS OF q -ARY SC-LDPC CODE ENSEMBLES ON THE BIAWGNC

A. q -ary Protograph EXIT Analysis on the BIAWGNC

We use the q -ary protograph EXIT (PEXIT) algorithm presented in [6] to analyze the FSD thresholds of protograph-based q -ary SC-LDPC code ensembles on the BIAWGNC, assuming that the binary image of a codeword is transmitted using BPSK modulation, and we extend it in a similar fashion to the q -ary WD-PDE algorithm to obtain WD thresholds for the BIAWGNC. Similar to the q -ary PDE analysis on the BEC, the q -ary PEXIT analysis is also a BP algorithm performed on a protograph, where the messages now represent *mutual information* (MI) values, a model obtained by approximating the distribution of the log-likelihood ratio messages in BP

decoding as (jointly) Gaussian. The thresholds are obtained by determining the smallest signal-to-noise ratio E_b/N_0 (in dB) for which decoding is successful, i.e., the smallest value of E_b/N_0 such that the *a-posteriori* MI between each variable node and a corresponding codeword symbol goes to 1 as the number of iterations goes to infinity.

B. Numerical Results

Our observations and conclusions made regarding the WD thresholds of q -ary SC-LDPC code ensembles on the BIAWGNC are similar to those made for the BEC. As a result, only a few examples are given here.

Fig. 14(a) compares the FSD thresholds of the $(2, 4, m)$ and $(3, 6, m)$ ensembles on the BIAWGNC.⁶ $\mathcal{C}_1(3, 6, m, 3)$ and $\mathcal{C}_1(3, 6, m, 1)$ have the same FSD thresholds for all m , which are almost identical to those of $\mathcal{C}_1(3, 6, m, 2)$, so only $\mathcal{C}_1(3, 6, m, 2)$ is shown to represent the $w = 1$ code ensembles and to compare with $\mathcal{C}_2(3, 6, m)$. Fig. 14(b) shows the WD thresholds of $\mathcal{C}_1(3, 6, m, 2)$ when $W = 3$ and 5. Both subfigures illustrate behavior similar to the BEC results presented in Section III-B. To summarize, small gains are observed for $\mathcal{C}_1(2, 4, m)$ compared to $\mathcal{B}(2, 4, m)$ until the finite field size gets large, whereas (numerically) capacity achieving WD thresholds that are significantly better than the corresponding block code thresholds are observed for both $\mathcal{C}_1(3, 6, m, 2)$ and $\mathcal{C}_2(3, 6, m)$. Again, $\mathcal{C}_1(3, 6, m, 2)$ turns out to be particularly well suited for WD; for $m = 5$ and $W = 5$, the WD threshold is essentially at capacity.

V. DECODING LATENCY AND DECODING COMPLEXITY

This section considers the tradeoff between two critical decoding properties of q -ary spatially coupled LDPC code ensembles:

- 1) *Latency*: measured as the number of bits that must be received before decoding can begin, and
- 2) *Complexity*: measured as the number of decoding operations required per information bit.

Our focus is the ensemble average behavior on the BEC when windowed decoding is used; different cases are compared on the same BEC, so that the tradeoff between decoding latency

⁶Due to computational limitations, the BIAWGNC thresholds were calculated only up to $m = 8$. However, as stated by Uchikawa *et al.* in [26], it is reasonable to assume that the BIAWGNC thresholds for $m = 9$ and 10 are consistent with the corresponding BEC results.

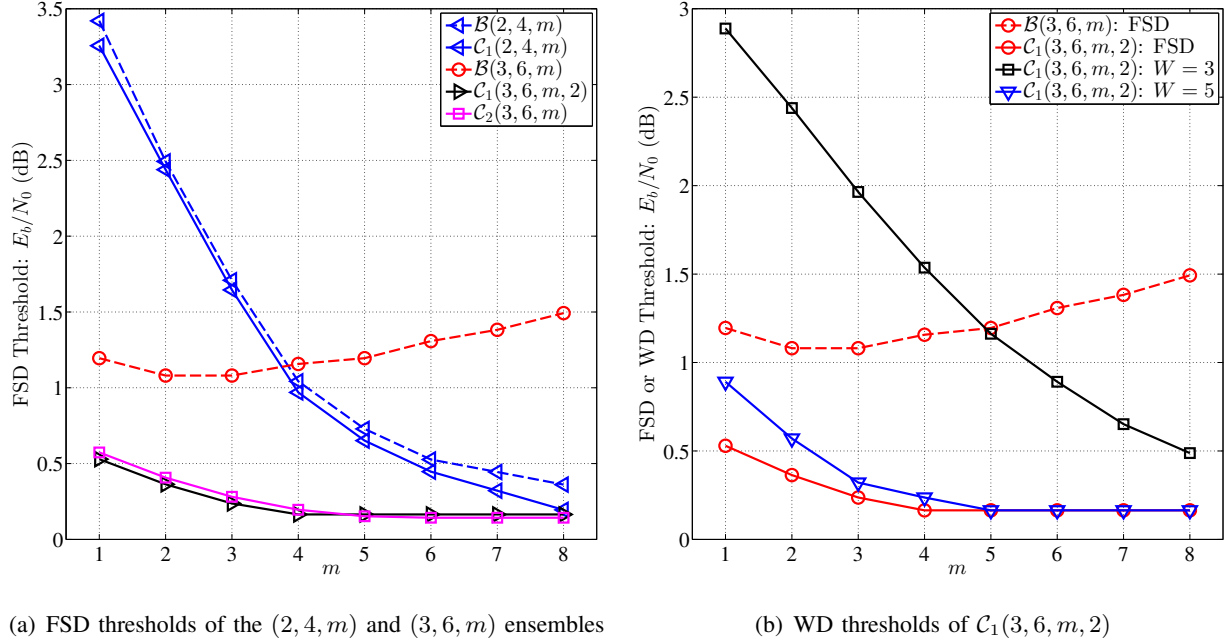


Fig. 14. FSD thresholds of the $(2, 4, m)$ and $(3, 6, m)$ ensembles and WD thresholds of $\mathcal{C}_1(3, 6, m, 2)$ on the BIAWGNC.

and decoding complexity can be examined. We use the q -ary WD-PDE algorithm (for WD) and the q -ary PDE algorithm (for FSD) in order to obtain our results, i.e., we consider an infinite lifting factor M used for the ensemble construction, thereby removing the effect of M from the latency-complexity tradeoffs. This allows us to get a general picture of the latency-complexity tradeoffs associated with a particular code ensemble, rather than analyzing specific codes, which can then be used to guide the design of practical, finite-length protograph-based q -ary SC-LDPC codes, especially when there is a limit on decoding latency.

In the remainder of this section, we focus on the $(J, 2J)$ SC-LDPC code ensembles with $k = 2$ and coupling width $w = 1$ previously discussed in Section III-B; however, similar results can be obtained for other code ensembles as well.

A. Decoding Latency

For a q -ary SC-LDPC code constructed as described in Section II-C, the decoding latency (normalized by M) of WD is given by kmW measured in *bits*, where we assume that the binary image of a codeword is transmitted, so each $\text{GF}(q)$ symbol contains m bits. For $k = 2$, the latency is proportional to the product of m and W . In the numerical results presented in this

section, we use

$$T_{\text{SC}} = 2mW \quad (35)$$

to represent the latency of WD for a q -ary SC-LDPC code ensemble. Also, FSD can be viewed as a special case of WD for which $W = L + w$, where L is the coupling length, so the corresponding latency can also be obtained using (35).

B. Decoding Complexity

As stated in [3] and the references therein, the decoding complexity of q -ary LDPC codes using the sum-product algorithm based on the fast Fourier transform can be summarized as follows:

- One check-to-variable (c-to-v) update requires $\mathcal{O}(q \log q)$ operations, and
- One variable-to-check (v-to-c) update requires $\mathcal{O}(q)$ operations.

We define the order of decoding complexity as the number of operations required per *information* bit, which is a fraction $1/(R_L m k M L)$ of the total number of operations for all the c-to-v and v-to-c updates during the decoding process, where R_L is the design rate. That is,

$$\begin{aligned} \text{Order of Decoding complexity} &= \mathcal{O} \left(\frac{J(q + q \log_2 q) k M \sum_{t=1}^L l_t}{R_L m k M L} \right) \\ &= \mathcal{O} \left(\frac{J 2^m (m + 1) \sum_{t=1}^L l_t}{R_L m L} \right), \end{aligned} \quad (36)$$

where l_t is the number of iterations involving updates of variables at time instant t ($1 \leq t \leq L$), which can be easily tracked during the decoding process. As previously mentioned, we let $L = 100$.

Although (36) is derived for BP decoding of finite-length SC-LDPC codes, we use it for our ensemble-average complexity analysis as well. For a particular q -ary SC-LDPC code ensemble, the erasure rate of the BEC is chosen to be no greater than the WD threshold of the ensemble, so q -ary WD-PDE is guaranteed to decode successfully. As the algorithm iterates, the number of c-to-v and v-to-c updates at each time instant is tracked via l_t , and then the order of decoding complexity is calculated.

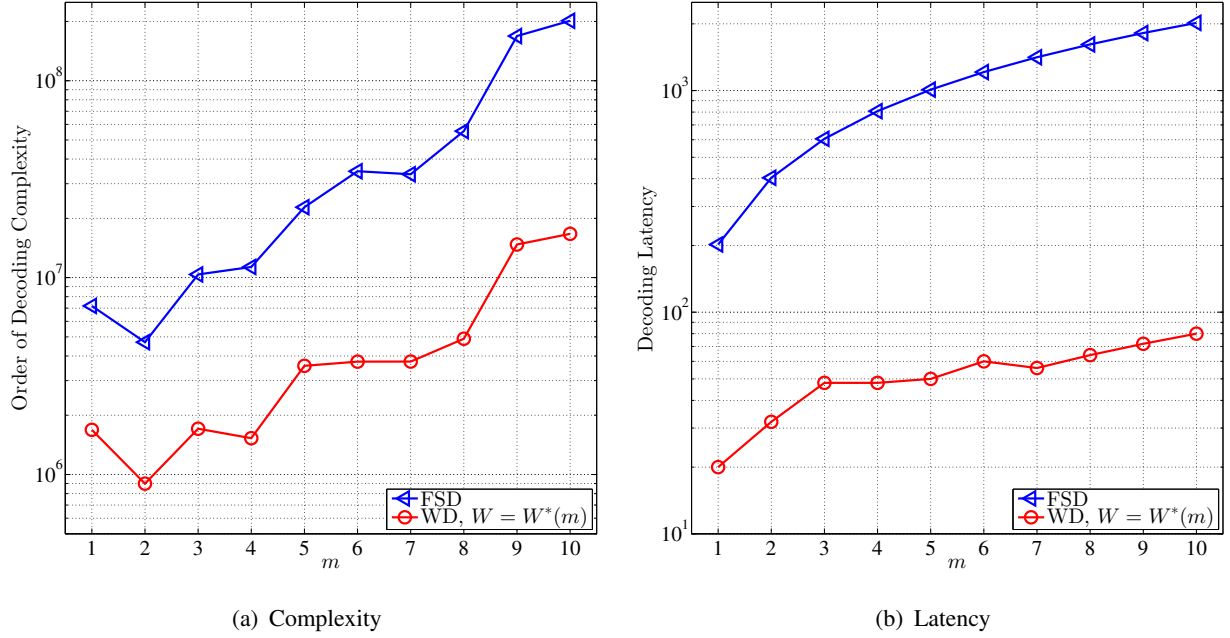


Fig. 15. WD vs. FSD for $\mathcal{C}_1(3, 6, m, 2)$: comparison of (a) decoding complexity, and (b) decoding latency.

C. Numerical Results

1) *WD vs. FSD, for the same decoding threshold*: Fig. 15 uses $\mathcal{C}_1(3, 6, m, 2)$ as an example to illustrate why WD is preferred to FSD for q -ary SC-LDPC code ensembles.

For each m , FSD is compared to WD with $W = W^*(m)$, where we recall that $W^*(m)$ is the minimum window size that provides the best achievable WD threshold $\hat{\epsilon}_{WD}(m)$. Here, $W^*(m) = 10, 8, 8, 6, 5, 5, 4, 4, 4$, and 4 for $m = 1$ to 10 . Here also the WD threshold $\hat{\epsilon}_{WD}(m)$ equals the FSD threshold $\epsilon^*(m)$ for $\mathcal{C}_1(3, 6, m, 2)$ for all m , i.e., these two cases have the same decoding threshold, and we set the channel erasure rate to this threshold, i.e., $\epsilon = \epsilon^*(m)$. From Fig. 15(a), we see that WD saves approximately 75% to 90% in decoding complexity compared to FSD, as m ranges from 1 to 10; the larger the finite field size, the more the savings in complexity.

Moreover, as shown in Fig. 15(b), WD also has a significant advantage in reducing decoding latency: the decoding latency of WD is only about 10% of FSD when $m = 1$ ($q = 2$) and only about 4% of FSD when $m = 10$ ($q = 1024$), i.e., the larger the finite field size, the more decoding latency is saved.

To summarize, WD is preferred to FSD for decoding q -ary SC-LDPC codes because the

former provides large savings in both decoding complexity and decoding latency, due to the fact that, unlike FSD, WD is localized to include only a small portion of the parity-check matrix. Also, by choosing the window size appropriately, these savings incur no loss in threshold.

2) *WD complexity as a function of m and W , with equal latency:* Decoding latency is calculated by $2mW$, so if mW is fixed, there can be multiple (m, W) pairs that satisfy a latency constraint. Again, using the $\mathcal{C}_1(3, 6, m, 2)$ ensemble as an example, Table I shows the order of decoding complexity of different (m, W) pairs, when $2mW$ is fixed at 24, 40, 48, 60, 80, and 120; the third column is for a BEC with $\epsilon = 0.488$, while the fourth column is for $\epsilon = 0.44$. For each ϵ , the smallest decoding complexity for a particular mW value is marked in boldface, corresponding to the most attractive (m, W) pair for that particular decoding latency.

The channel erasure rate $\epsilon = 0.488$ is within approximately 0.1% of the best-achievable binary WD threshold of $\mathcal{C}_1(3, 6, m = 1, 2)$ and 2.5% from channel capacity.⁷ As a result, when $m = 1$, a large number of iterations is required to achieve decoding success using the q -ary WD-PDE algorithm. On the other hand, larger values of m (and as a result, smaller values of W), for example, $m = 2, 3$, and 4, show significant reductions in decoding complexity (one to two orders of magnitude), since the WD thresholds for the corresponding (m, W) pairs are larger; in fact, the smallest decoding complexity is achieved when m is either 2 or 3 for all the decoding latencies examined in Table I.

q -ary SC-LDPC codes may still provide benefits compared to their binary counterparts even at lower channel erasure rates. For example, in the fourth column of Table I, $\epsilon = 0.44$ is approximately 10% from the best achievable binary WD threshold of $\mathcal{C}_1(3, 6, m, 2)$ and 12% from channel capacity. Here, we see that $m = 2$ has lower decoding complexity than $m = 1$ for all decoding latencies and achieves the smallest complexity in all cases, although the gains compared to the $\epsilon = 0.488$ case are not large. Eventually, the advantage of 4-ary codes compared to binary codes disappears as ϵ decreases further; nevertheless, Table I suggests that, for near-capacity performance requirements with a constraint on decoding latency, one should consider q -ary SC-LDPC codes as alternatives to binary codes.

⁷For a fixed value of mW , not all possible (m, W) pairs can guarantee decoding success for this channel erasure rate. For example, when $mW = 12$, $m = 3$ and $W = 4$ results in a threshold below $\epsilon = 0.488$.

TABLE I
ORDER OF DECODING COMPLEXITY OF $\mathcal{C}_1(3, 6, m, 2)$

Decoding latency $2mW$	(m, W)	Order of decoding complexity	
		$\epsilon = 0.488$	$\epsilon = 0.44$
24	(1, 12)	4.55×10^5	1.14×10^3
	(2, 6)	9.25×10^3	1.03×10^3
40	(1, 20)	7.18×10^5	1.77×10^3
	(2, 10)	1.32×10^4	1.37×10^3
	(4, 5)	1.54×10^4	2.77×10^3
	(5, 4)	2.62×10^4	4.92×10^3
48	(1, 24)	8.38×10^5	2.07×10^3
	(2, 12)	1.57×10^4	1.62×10^3
	(3, 8)	1.36×10^4	2.05×10^3
	(4, 6)	1.76×10^4	3.07×10^3
	(6, 4)	4.43×10^4	8.96×10^3
	(8, 3)	2.06×10^5	3.12×10^4
60	(1, 30)	1.00×10^6	2.49×10^3
	(2, 15)	1.92×10^4	1.98×10^3
	(3, 10)	1.68×10^4	2.51×10^3
	(5, 6)	3.23×10^4	5.91×10^3
	(6, 5)	5.15×10^4	9.77×10^3
	(10, 3)	5.15×10^5	1.12×10^5
80	(1, 40)	1.23×10^6	3.09×10^3
	(2, 20)	2.48×10^4	2.55×10^3
	(4, 10)	2.84×10^4	4.88×10^3
	(5, 8)	4.24×10^4	7.75×10^3
	(8, 5)	1.91×10^5	3.67×10^4
	(10, 4)	5.85×10^5	1.13×10^5
120	(1, 60)	1.54×10^6	3.94×10^3
	(2, 30)	3.46×10^4	3.58×10^3
	(3, 20)	3.16×10^4	4.71×10^3
	(4, 15)	4.13×10^4	7.10×10^3
	(5, 12)	6.21×10^4	1.13×10^4
	(6, 10)	9.95×10^4	1.88×10^4
	(10, 6)	8.63×10^5	1.66×10^5

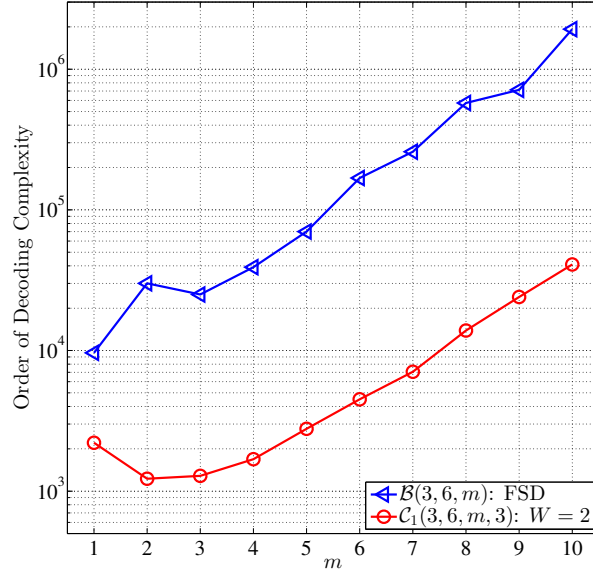


Fig. 16. Comparison of decoding complexity: $\mathcal{B}(3, 6, m)$ using FSD and $\mathcal{C}_1(3, 6, m, 3)$ using WD with $W = 2$.

3) $\mathcal{B}(J, 2J, m)$ vs. $\mathcal{C}_1(J, 2J, m, 3)$, with equal latency: We now compare the WD complexity of the $\mathcal{C}_1(J, 2J, m, 3)$ ensembles (with $\mathbf{B}_0 = \begin{bmatrix} J-1 & J-1 \end{bmatrix}$ and $\mathbf{B}_1 = \begin{bmatrix} 1 & 1 \end{bmatrix}$) with $W = 2 = w + 1$ to the FSD complexity of the $\mathcal{B}(J, 2J, m)$ ensembles defined by

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_0 & \mathbf{B}_1 \\ \mathbf{B}_1 & \mathbf{B}_0 \end{bmatrix}. \quad (37)$$

Similar to the derivation of (35), the FSD (normalized) latency of $\mathcal{B}(J, 2J, m)$ is

$$T_{\text{BC}} = 4m, \quad (38)$$

the same as $T_{\text{SC}} = 2mW = 4m$ for $\mathcal{C}_1(J, 2J, m, 3)$ with $W = 2$, i.e., the decoding latencies are equal.

The orders of decoding complexity for $J = 3$ are illustrated in Fig. 16. For each m , the channel erasure rate is chosen as the FSD threshold $\epsilon^*(m)$ of $\mathcal{B}(3, 6, m)$, which is smaller than the WD threshold $\epsilon_{\text{WD}}^*(m, W)$ of $\mathcal{C}_1(3, 6, m, 3)$ with $W = 2$. Similar results can be obtained for $J = 4$ and $J = 5$ as well. For $w = 1$, using $W = 2$ results in the smallest possible decoding latency, so Fig. 16 suggests that, even under a very tight latency constraint, q -ary SC-LDPC code ensembles with type 3 spreading still provide a significant reduction in decoding complexity compared to their block code counterparts. For a comparison of finite-length q -ary SC-LDPC codes and q -ary

LDPC-BCs, where the lifting factor M can be varied to achieve various tradeoffs between error probability, complexity, and latency, we refer the reader to [32].

VI. CONCLUSIONS

This paper proposes design rules for q -ary spatially coupled LDPC codes suitable for latency-constrained applications. The design rules are based on an analysis of the windowed decoding thresholds of various protograph-based (J, K) -regular q -ary SC-LDPC code ensembles for both the binary erasure channel and the BPSK-modulated additive white Gaussian noise channel. In particular, we show that mixing \mathbf{E}_A and \mathbf{E}_B edge spreadings to construct q -ary SC-LDPC code ensembles results in near-capacity WD thresholds when both the finite field size q and the window size W are relatively small, and that the balance between these two types of spreading depends on the degree distribution and the threshold requirements.

By tracking the number of density evolution update operations needed for decoding success of a q -ary SC-LDPC code ensemble for fixed channel conditions, we also demonstrate that WD is superior to FSD in both decoding complexity and decoding latency. Finally, when operation close to the binary SC-LDPC code ensemble threshold is required, we show that codes from q -ary SC-LDPC code ensembles provide significant reductions in decoding complexity compared to binary codes for the same decoding latency.

REFERENCES

- [1] M. C. Davey and D. J. C. MacKay, "Low-density parity check codes over $GF(q)$," *IEEE Communication Letters*, vol. 2, no. 6, pp. 165–167, June 1998.
- [2] L. Barnault and D. Declercq, "Fast decoding algorithm for LDPC over $GF(2^q)$," in *IEEE Information Theory Workshop*, pp. 70–73, Paris, France, Apr. 2003.
- [3] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, and P. Urard, "Low-complexity decoding for non-binary LDPC codes in high order fields," *IEEE Transactions on Communications*, vol. 58, no. 5, pp. 1365–1375, May 2010.
- [4] Erbao Li, D. Declercq, and K. Gunnam, "Trellis-based extended min-sum algorithm for non-binary LDPC codes and its hardware structure," *IEEE Transactions on Communications*, vol. 61, no. 7, pp. 2600–2611, July 2013.
- [5] A. Bennatan and D. Burshtein, "Design and analysis of nonbinary LDPC codes for arbitrary discrete-memoryless channels," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 549–583, Feb. 2006.
- [6] L. Dolecek, D. Divsalar, Y. Sun, and B. Amiri, "Non-binary protograph-based LDPC codes: Enumerators, analysis, and designs," *IEEE Transactions on Information Theory*, vol. 60, no. 7, pp. 3913–3941, July 2014.
- [7] J. Thorpe, "Low-density parity-check (LDPC) codes constructed from protographs," JPL IPN Progress Report 42-154, Aug. 2003.

- [8] A. E. Pusane, R. Smarandache, P. O. Vontobel, and D. J. Costello, Jr., "Deriving good LDPC convolutional codes from LDPC block codes," *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 835–857, Feb. 2011.
- [9] T. J. Richardson and R. L. Urbanke, *Modern coding theory*. Cambridge University Press, 2008.
- [10] D. Divsalar, S. Dolinar, C. Jones, and K. Andrews, "Capacity-approaching protograph codes," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 6, pp. 876–888, Aug. 2009.
- [11] G. Liva and M. Chiani, "Protograph LDPC codes design based on EXIT analysis," *IEEE Global Telecommunications Conference*, pp. 3250–3254, Washington, U.S., Nov. 2007.
- [12] A. Jiménez Felström and K. Sh. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrices," *IEEE Transactions on Information Theory*, vol. 45, no. 6, pp. 2181–2191, Sept. 1999.
- [13] M. Lentmaier, A. Sridharan, D. J. Costello, Jr., and K. Sh. Zigangirov, "Iterative decoding threshold analysis for LDPC convolutional codes," *IEEE Transactions on Information Theory*, vol. 56, no. 10, pp. 5274–5289, Oct. 2010.
- [14] S. Kudekar, T. J. Richardson, and R. L. Urbanke, "Threshold saturation via spatial coupling: Why convolutional LDPC ensembles perform so well over the BEC," *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 803–834, Feb. 2011.
- [15] S. Kudekar, T. Richardson, and R. Urbanke, "Spatially coupled ensembles universally achieve capacity under belief propagation," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 7761–7813, Dec. 2013.
- [16] G. Miller and D. Burshtein, "Bounds on the maximum-likelihood decoding error probability of low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, no. 7, pp. 2696–2710, Nov. 2001.
- [17] M. Lentmaier, G. P. Fettweis, K. S. Zigangirov, and D. J. Costello, "Approaching capacity with asymptotically regular LDPC codes," *Information Theory and Application Workshop*, San Diego, U.S., Feb. 2009.
- [18] D. G. M. Mitchell, M. Lentmaier, and D. J. Costello, "New families of LDPC block codes formed by terminating irregular protograph-based LDPC convolutional codes," *IEEE International Symposium on Information Theory*, pp. 824–828, Austin, U.S., June 2010.
- [19] Z. Si, R. Thobaben, and M. Skoglund, "Bilayer LDPC convolutional codes for half-duplex relay channels," *IEEE Transactions on Communications*, vol. 1, no. 8, pp. 3086–3099, Aug. 2013.
- [20] K. Kasai and K. Sakaniwa, "Spatially-coupled MacKay-Neal codes and Hsu-Anastasopoulos codes," 2013. [Online]. Available: <http://arxiv.org/pdf/1102.4612v3.pdf>.
- [21] S. Kudekar and K. Kasai, "Threshold saturation on channels with memory via spatial coupling," *IEEE International Symposium on Information Theory*, pp. 2562–2566, St. Petersburg, Russia, Aug. 2011.
- [22] S. Kudekar and K. Kasai, "Spatially coupled codes over the multiple access channel," *IEEE International Symposium on Information Theory*, pp. 2816–2820, St. Petersburg, Russia, Aug. 2011.
- [23] P. S. Nguyen, A. Yedla, H. D. Pfister, and K. R. Narayanan, "Threshold saturation of spatially-coupled codes on intersymbol-interference channels," *IEEE International Conference on Communications*, pp. 2181–2186, Ottawa, Canada, June 2012.
- [24] H. Uchikawa, K. Kasai, and K. Sakaniwa, "Spatially coupled LDPC codes for decode-and-forward in erasure relay channel," *IEEE International Symposium on Information Theory*, pp. 1474–1478, St. Petersburg, Russia, July 2011.
- [25] D. G. M. Mitchell, M. Lentmaier, and D. J. Costello, Jr., "Spatially coupled LDPC codes constructed from protographs," submitted to the *IEEE Transactions on Information Theory*, 2014. [Online]. Available: <http://arxiv.org/abs/1407.5366>.
- [26] H. Uchikawa, K. Kasai, and K. Sakaniwa, "Design and performance of rate-compatible non-binary LDPC convolutional codes," 2011. [Online]. Available: <http://arxiv.org/pdf/1010.0060v2.pdf>

- [27] A. Piemontese, A. Graell i Amat, and G. Colavolpe, "Nonbinary spatially-coupled LDPC codes on the binary erasure channel," in *IEEE International Conference on Communications*, pp. 3270–3274, Budapest, Hungary, June 2013.
- [28] I. Andriyanova and A. Graell i Amat, "Threshold saturation for nonbinary SC-LDPC codes on the binary erasure channel," 2013. [Online]. Available: <http://arxiv.org/abs/1311.2003/>
- [29] A. R. Iyengar, M. Papaleo, P. H. Siegel, J. K. Wolf, A. Vanelli-Coralli, and G. E. Corazza, "Windowed decoding of protograph-based LDPC convolutional codes over erasure channels," *IEEE Transactions on Information Theory*, vol. 58, no. 4, pp. 2303–2320, Apr. 2012.
- [30] M. Lentmaier, M. M. Prenda, and G. P. Fettweis, "Efficient message passing scheduling for terminated LDPC convolutional codes," in *IEEE International Symposium on Information Theory*, pp. 1826–1830, St. Petersburg, Russia, Aug. 2011.
- [31] V. Rathi and R. L. Urbanke, "Density evolution, thresholds and the stability condition for non-binary LDPC codes," *IEEE Proceedings – Communications*, vol. 152, no. 6, pp. 1069–1074, Dec. 2005.
- [32] K. Huang, D. G. M. Mitchell, L. Wei, X. Ma, and D. J. Costello, Jr., "Performance comparison of non-binary LDPC block and spatially coupled codes," submitted to *IEEE Transactions on Communications*, 2014. [Online]. Available: <http://arxiv.org/abs/1408.2621>.